IRMA™

International User's Manual

Digital Communications Associates, Inc.

dca®

IRMA

International
User's Manual

Digital
Communications
Associates, Inc.

dca®

In all correspondence with Digital Communications Associates, Inc. regarding this publication, please refer to: Document Number: MIC-016B 7/85.

WARNING: This equipment generates, uses, and can radiate radio frequency energy and if not installed and used in accordance with the instructions manual, may cause interference to radio communications. It has been tested and found to comply with the limits for a class A computing device pursuant to subpart J of Part 15 of FCC Rules, which are designed to provide reasonable protection against such interference when operated in a commercial environment. Operation of this equipment in a residential area is likely to cause interference, in which case the user, at his own expense, will be required to take whatever measures may be required to correct the interference.

Table of Contents

Chapter 1 Introduction

Chapter 2 Installation

Chapter 3 Using IRMA and the Terminal Emulator

Chapter 4 *File Transfer*

Chapter 5 Technical Reference

Appendix A *Error Conditions*

Appendix B *Using GENX*

Appendix C BASICA Subroutines

Customer Support Information

Trademarks and Registered Names

IBM is a registered trademark of International Business Machines Corporation.

APL-I, VM/CMS, MVS/TSO are trademarks of International Business Machines Corporation.

APL*PLUS/PC is a trademark of STSC, Inc.

IRMA, IRMA*link FT78X*, and IRMA*link FT78T* are trademarks of Digital Communications Associates, Inc.

Eagle is a trademark of Eagle Computers, Inc.

COMPAQ is a trademark of COMPAQ Computer Corporation.

WORDSTAR is a registered trademark of Micro Pro International Corporation.

Microsoft is a registered trademark of Microsoft Corporation.

DCA is a registered trademark of Digital Communications Associates, Inc.

Chapter 1

Introduction

Product Description

IRMA,™ a member of DCA's® family of micro-to-mainframe links provides the IBM® PC, PC XT, or PC AT with a direct high-speed link to IBM 3270 networks. IRMA offers a coaxial cable connection to IBM 3274, 3276, or Integral Terminal controllers with Type "A" adapters. It can operate with both channel attached and remote (BSC or SNA/SDLC linked) 3274 controllers.

IRMA includes a printed circuit board (card) that fits into any available slot in IBM PC's and a software package that consists of a 3278/79 Terminal Emulator program, called E78, and two file transfer utilities for TSO and CMS environments. Also included in the software are BASICA subroutines which provide keystroke and field access from a BASICA program to the 3270 controller. These routines are useful in developing programs for automatic data transfer.

The 3278/79 Terminal Emulator provides all users with the features of a 3278 type terminal. When using the emulator, the keys found on a 3278 terminal are available on the PC keyboard. However, due to the differences in the layouts of the two keyboards, some key positions change. Keyboard charts and decals are provided with IRMA to aid the user with these key changes.

IRMA's operation requires no mainframe software or special network configuration. There is no additional software required to handle the 3270 protocol other than the on-board terminal emulator software.

Thus, the IBM PC with an IRMA installed can immediately serve as either an independent workstation or as a full-functioned 3278/79 workstation.

IRMA Features

Complete 3278 Emulation

IRMA enables the PC to fully emulate a 3278 monochrome or 3279 color terminal with up to 3440-character display and 80-character status indicator line—full 3278/79 emulation, plus stand-alone PC processing.

File Transfer

IRMA's easy-to-use file transfer program, with its HELP function and question-and-answer format, provides all the information required for successful transfer of files under the TSO or CMS IBM mainframe software packages.

Emulator Program Customization

GENX, a menu-driven program, allows you to tailor IRMA's features and function modes to your individual system requirements. GENX may be used to create customized versions of the E78 Terminal Emulator program with specialized keyboard, color, and communication defaults.

Keyboard Emulation

IRMA users have the ability to redefine the PC keyboard to functionally correspond to the 3278 typewriter-style keyboard. Alternate (Alt) and Control (Ctrl) key functions allow easy emulation of PF and PA keys. IRMA supports all 16 possible IBM keyboard types, including "reserved" types used by IBM for custom keyboards. A color-coded keyboard chart is provided for easy identification of all normal and special key functions.

Keyboard Reconfiguration

IRMA users have the ability to reconfigure the keyboard. All E78 Terminal Emulator program keyboard sequences are controlled by simple tables which may be user-modified using the GENX program feature. Since changing the keyboard requires changing the IRMA tables, this task should be handled by an experienced programmer. An additional document that explains this procedure in detail can be requested from either DCA's Customer Support or Technical Support Group.

Instant Emulator Availability

IRMA's emulator program can be made resident upon execution. With this feature, an "auto resident" E78 can be placed in the AUTOEXEC.BAT file to make terminal emulation instantly available at machine startup.

On-Board Screen Buffer

IRMA maintains a complete screen buffer in memory. The buffer is accessible from user programs that run in the PC itself, allowing you to develop your own selective access to corporate databases, retrieving data from the mainframe, and returning the modified screen.

Independent, Concurrent Operation

IRMA's on-board high-speed microprocessor handles the 3270 protocol independently of the PC's 8088 processor. This permits an IBM PC or PC XT to be attached and on-line as a 3278/79 terminal while also allowing full stand-alone processing activities by the PC.

Attribute/EAB Support

IRMA supports Attribute and Extended Attribute (EAB) characters for field-oriented screens.

Light Pen Support

IRMA provides light pen support for applications requiring the IBM "Selector Pen". This feature requires that the E78 Terminal Emulator program be used with the IBM Color Display Adapter, and a light pen which connects to the display adapter (such as the one sold by FTG Data Systems, Stanton, California).

ASYNC Character Support

IRMA allows data entry to IBM mainframes using character serial devices attached to the PC RS-232 ASYNC card on COM1: The interface supports such options as barcode or OCR readers and touch input screens (such as the screen manufactured by Touch Technology, Annapolis, Maryland).

APL Support

IRMA supports the IBM APL-I character set and keyboard. This support requires a display adapter equipped with DCA or STSC APL*PLUS/PC™ character generation ROM. (For more information concerning the DCA APL option, contact DCA Customer Support at 1-800-631-4171. In Georgia call 1-404-442-4470.)

43-Line Screen Support

IRMA's E78 Terminal Emulator program may be configured to appear to the mainframe as a 3278 MOD 2 (24x80), MOD 3 (32x80), or MOD 4 (43x80), or 3279 (with full seven-color support). Screens longer than 24 lines are handled by scrolling key functions and an automatic cursor tracking system. MOD 4 support is standard with Revision 1.06 (or higher) IRMA cards. Revision 1.05 IRMA cards can be modified to support MOD 4. To modify the 1.05 IRMA card, contact DCA's Customer Support at 1-800-631-4171 for the proper procedures.

PC 'Look-Similar' Support

IRMA is able to support PC "look-similars" using 8086 processors (such as the Eagle 1600). Also included is support for hybrid COLOR/MONOCHROME screens such as those used on the COMPAQ™ portable computer.

What to Do First

The IRMA package consists of the following items. Check to see that you have received the complete kit.

- IRMA documentation Package
- IRMA printed circuit board
- IBM PC compatible diskette containing the IRMA software
- IRMA keyboard decal kit
- Plastic card guide.

If any items are missing, contact your IRMA distributor.

Now turn to the Customer Support Information found at the end of the IRMA User's Manual and read the warranty information carefully. A DCA product registration card is also located in this section. Fill out this card and return it to DCA. This validates your IRMA product warranty and provides DCA with a means to keep you informed of any product updates that may occur.

Information About This Manual

The purpose of this manual is to provide you with complete instructions for installing, configuring, and using the IRMA product. This manual does not discuss details about data communications concepts, IBM 3270 terminal operations, or network application programs. If you are not familiar with these topics, refer to your systems supervisor or documentation relating to the specific subject of interest. IBM's publication, IBM GA27-2849, 3270 Information Display System Configuration, contains information about the 3270 family of terminals.

The IRMA User's Manual is divided into five chapters, three appendices, and a Customer Support section.

Chapter 1, "Introduction," describes the IRMA product's basic features and benefits. This chapter also provides a checklist of the IRMA package and a description of the information contained in each chapter.

Chapter 2, "Installation," presents step-by-step procedures for installing the IRMA printed circuit board and applying the decals to your PC keyboard. If you require any of IRMA's special features, such as APL support or support for the PC-compatibles (COMPAQ or EAGLE™), there are instructions for customizing the emulator software to accommodate these special needs.

Chapter 3, "Using IRMA and the Terminal Emulator," describes the operations and functions of the emulator. All of IRMA's special functions are described in this chapter. There is also a discussion of the 3270 system messages and how they appear on the PC.

Chapter 4, "File Transfer," presents two file transfer programs, one for TSO environments, and one for CMS environments. Contained in this chapter are the step-by-step procedures for using the transfer programs.

Chapter 5, "Technical Reference," presents the necessary information required for a programmer to access IRMA for complex, specialized software development.

Appendix A, "Error Conditions," includes error messages relating to IRMA and how to rectify simple problems that may arise in daily use.

Appendix B, "Using GENX," describes in detail the menu options used to customize the 3278 emulation software.

Appendix C, "BASICA Subroutines," discusses the use of the subroutines provided with the IRMA software. These routines can be used by an experienced BASICA programmer for developing programs for file transfer.

Customer Support Information contains the product warranty and registration card, service information, and a reader comment page.

How to Use This Manual

The IRMA User's Manual provides information for both the basic and advanced computer user. You may not need to read all of the user's guide to adequately use IRMA and the 3278 emulation program. Use the following chart to determine what information you need.

If you want to...	Read Chapters					Read Appendices		
	1	2	3	4	5	A	B	C
Install IRMA		x						
Customize 3278 Emulation software		x					x	
Use the emulator		x	x					
Use File transfer				x				
Develop Assembly Language Programs					x			x
Develop BASICA programs					x			x
Troubleshoot IRMA					x	x		

NOTE: E78 refers to the IRMA Terminal Emulator. Throughout the remainder of this manual E78 and IRMA Terminal Emulator software are synonymous.

Chapter 2

Installation

Introduction

IRMA's installation is broken into three phases. Phase one is the physical installation of the IRMA board. Phase two is the application of the keyboard decals. Phase three is specifying special options through a menu driven customization routine called GENX. Not all users need to use GENX. However, everyone should read the discussion on 'Special Options' found later in this chapter to determine whether you need to use GENX or not. Also included in this chapter are instructions for making backup copies of your IRMA diskette.

Items You Need to Install IRMA

If you are ready to install IRMA, check first to see that you have everything you will need to perform the installation. You must have the following items:

- Your IRMA circuit board.
- An IBM PC or a PC-compatible machine.
- A copy of the IBM PC Guide to Operations or any other manual that contains instructions for the removal and re-assembly of the PC cabinet, as well as instructions for installation of an option into the PC.

- Any tools suggested by your IBM manual for the operations described above.
- Small tool with one sharp end and one blunt end for applying the keyboard decals.

Cables and Connections

Now that you have gathered the equipment and tools you will need for the installation of the IRMA card and the decals, it is necessary to discuss the way your PC will be connected to the mainframe system. IBM uses coaxial cable to connect certain types of equipment such as printers and terminals to other types of equipment, such as a mainframe terminal controller. If your PC and IRMA are to communicate with the mainframe system, you must have a coaxial cable running into the work area where your PC is located. The other end of this coax cable must be attached to the coax port of your mainframe terminal controller. If your PC is replacing an IBM 3278 or 3279 terminal, this cable may already be present.

The terminal controller is able to communicate not only with terminals, but with other devices as well (such as a printer). The controller must understand what type of device is attached by cable to its port. Therefore, the controller cable port you will use with your PC and IRMA must be "configured" to work with a 3278 or 3279 terminal and a typewriter-style keyboard. If you will be using an existing cable that is presently, or was previously, attached to an IBM 3278/79 terminal, the controller port on the other end of the cable is probably configured correctly at this time.

If you are not sure if you have a coax cable in your work area, or if the existing cable requires configuration, see your MIS or other technical support personnel for help.

How to Install IRMA

If you have all the necessary equipment and tools, you are ready to install your IRMA board. Follow the instructions below.

1. Following the instructions outlined in your PC manual, remove the cover of the PC unit.

WARNING: Unplug the PC unit prior to the installation of any accessory card. To ensure safety, unplug the system unit line cord before removing the cover. Following installation, completely re-assemble the cabinet before applying power.

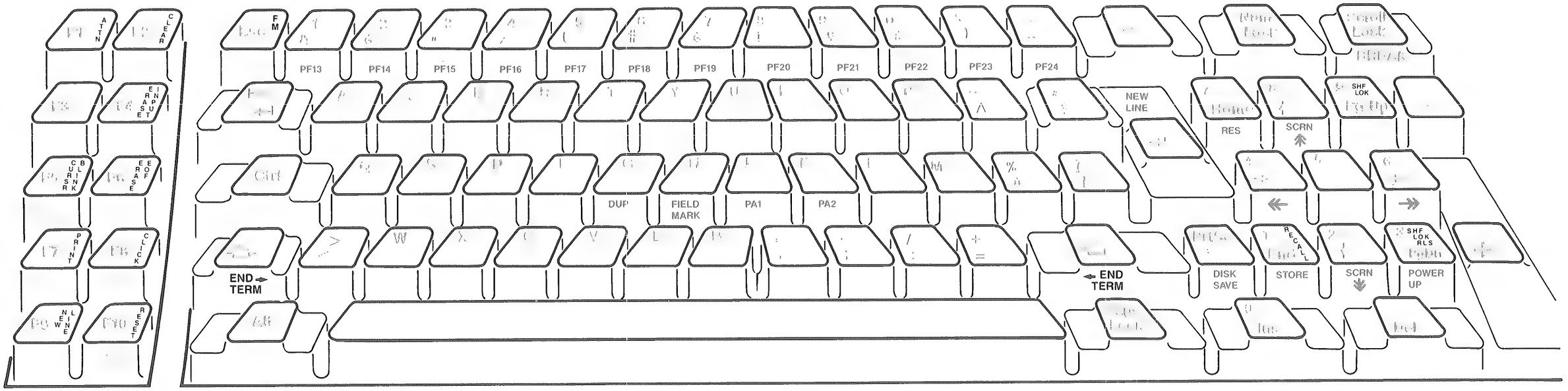
2. Locate an open expansion slot in the PC and install the IRMA card into this slot. For instructions detailing this procedure, refer to the "Options" section of your IBM PC Guide to Operations. IRMA installation requires no switch settings.
3. Following the instructions in your PC manual, re-assemble the PC cabinet. Be sure to re-attach all cables as instructed. The rear panel of your PC now has a coax connector projecting from the IRMA card you have just installed.
4. Attach the coaxial cable described in the cable instructions to the IRMA coaxial connector on the rear panel of your PC. To attach the cable, gently push and turn the connector 1/8th turn clockwise until it locks. If you should need to remove the cable, simply reverse this process, pushing gently and turning the connector 1/8th turn counter-clockwise.
5. Plug in your PC unit.
6. Your IRMA card is now physically installed. The next phase of the installation is to apply the key decals that are provided as part of the IRMA package.

Keyboard

The keyboard charts on the following pages are provided as a guide to the location of the keys and key functions available to you as an IRMA user. The chart uses a color code system that identifies normal PC key functions, 3278/79 key functions, and special IRMA functions. The chart may also be used as a guide to placement of the color-coded decals provided with your user's manual.

To apply your color-coded keyboard decals, see the appropriate fold-out keyboard chart and follow the instructions below.

1. Locate a sharp-tipped tool (small knife blade, nail file, or similar item), and choose a starting point on the chart.
2. Using the tip of the tool, carefully lift the decal by its top edge and position it onto the appropriate key in the location indicated on the keybord chart.
3. Using a fingertip or smooth blunt object, smooth the decal onto the key. Avoid touching the adhesive with your hands.
4. Repeat the process for each remaining key, being careful to exactly match the keyboard chart.



Belgian Keyboard

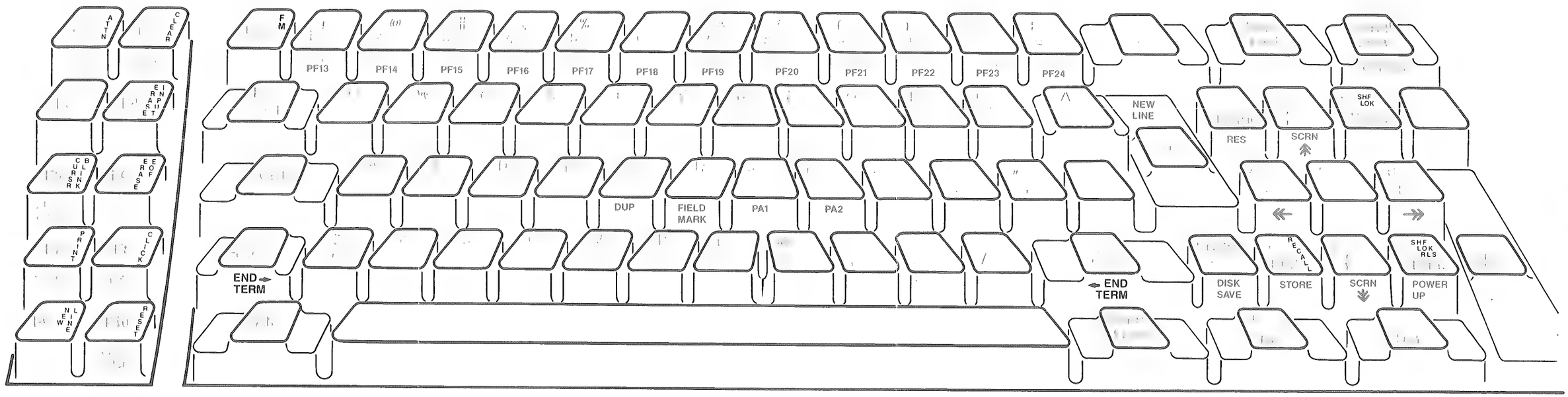
This keyboard chart is provided as a guide to the location of keys that invoke special IRMA functions. The chart may also be used as a guide to placement of the color-coded decals provided in the documentation package.

Legends in [ ] Print
These keys represent normal PC functions. Some of these functions may not be available while running the terminal emulator.

Legends in [ ] Print (Black on decals)
These keys are used to invoke special IRMA functions while running the terminal emulator. While in the emulator mode, these keys replace normal PC legends.

Legends in [ ] Print (Blue on decals)
These keys are used in combination with the Alternate (Alt) key to perform certain commands or functions. To invoke Alt functions, press the Alt key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PFI function, press simultaneously the Alt and 1 keys.

Legends in [ ] Print (Red on decals)
These keys are used in combination with the Control (Ctrl) key to perform certain commands or functions. To invoke Ctrl functions, press the Ctrl key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PF13 function, press simultaneously the Ctrl and 1 keys.



Canadian-French Keyboard

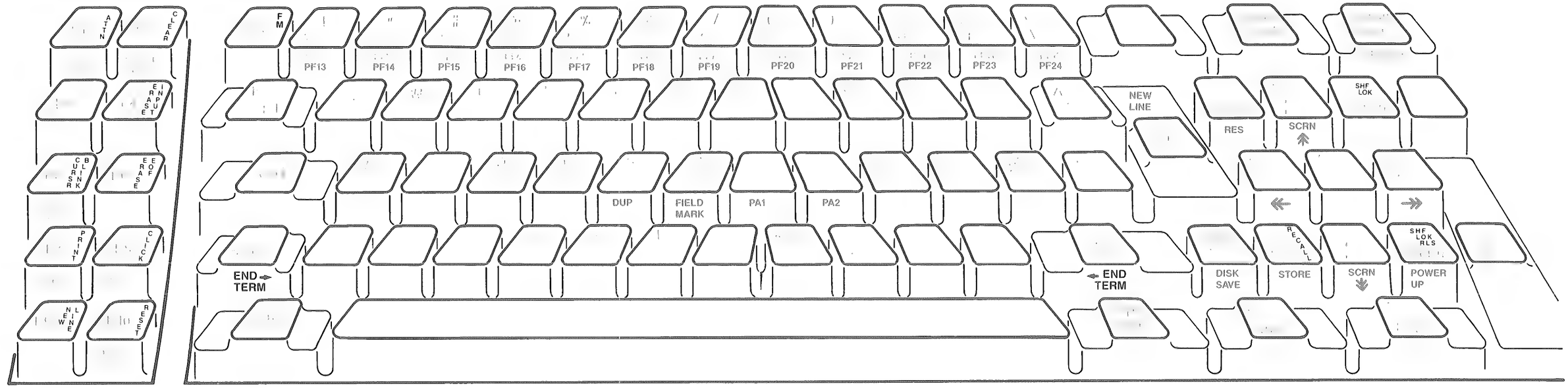
This keyboard chart is provided as a guide to the location of keys that invoke special IRMA functions. The chart may also be used as a guide to placement of the color-coded decals provided in the documentation package.

Legends in [redacted] Print
These keys represent normal PC functions. Some of these functions may not be available while running the terminal emulator.

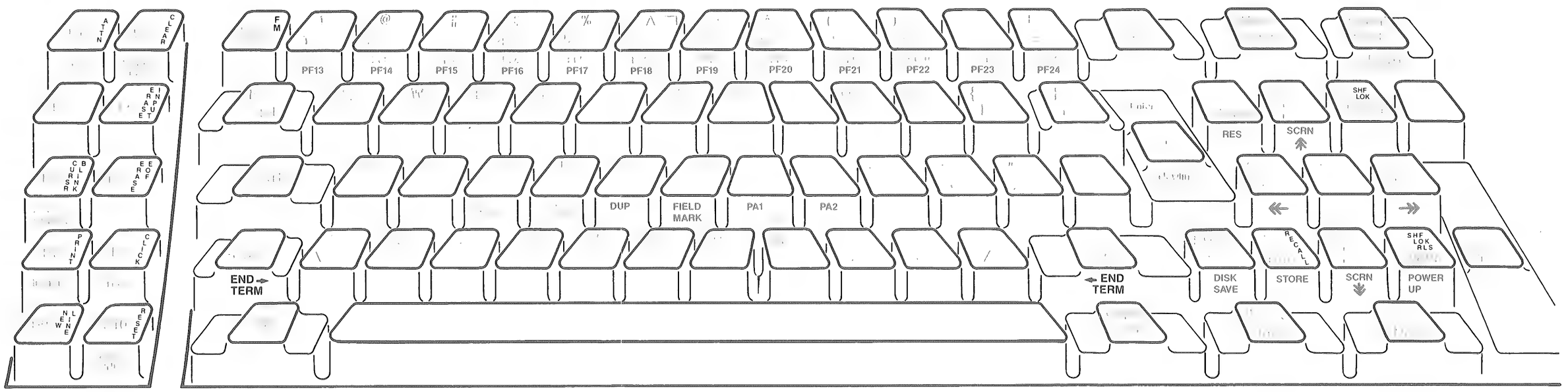
Legends in [redacted] Print (Black on decals)
These keys are used to invoke special IRMA functions while running the terminal emulator. While in the emulator mode, these keys replace normal PC legends.

Legends in [redacted] Print (Blue on decals)
These keys are used in combination with the Alternate (Alt) key to perform certain commands or functions. To invoke Alt functions, press the Alt key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PFI function, press simultaneously the Alt and 1 keys.

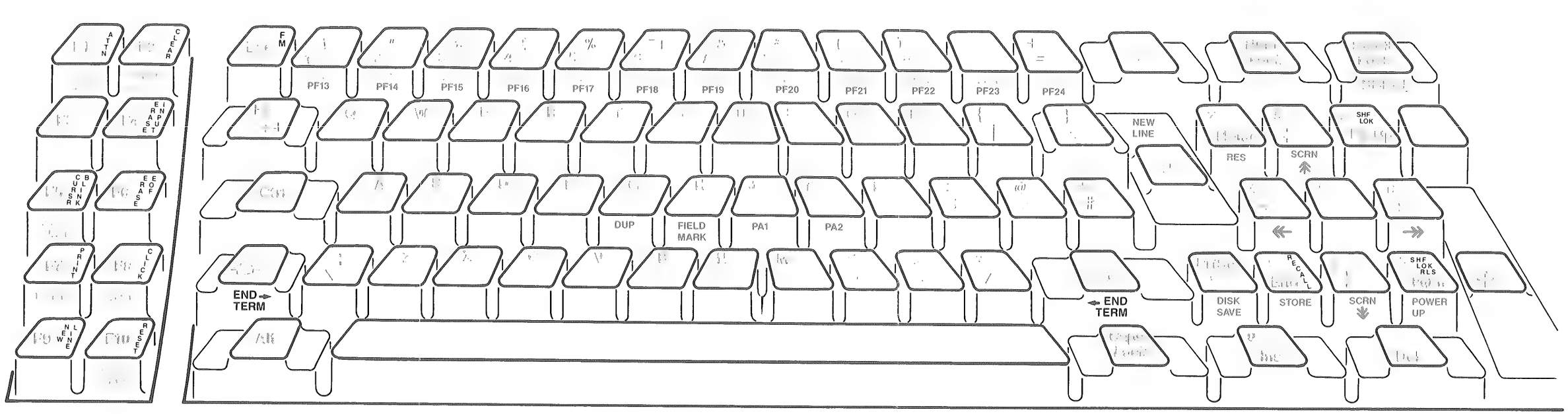
Legends in [redacted] Print (Red on decals)
These keys are used in combination with the Control (Ctrl) key to perform certain commands or functions. To invoke Ctrl functions, press the Ctrl key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PF13 function, press simultaneously the Ctrl and 1 keys.



Danish Keyboard



English (Domestic) Keyboard



English (UK) Keyboard

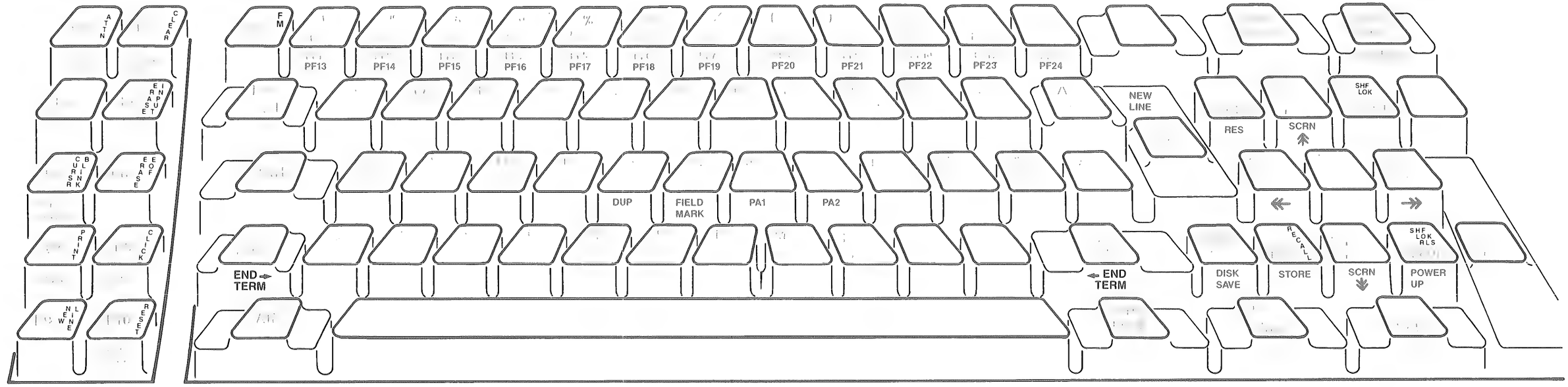
This keyboard chart is provided as a guide to the location of keys that invoke special IRMA functions. The chart may also be used as a guide to placement of the color-coded decals provided in the documentation package.

Legends in Print
These keys represent normal PC functions. Some of these functions may not be available while running the terminal emulator.

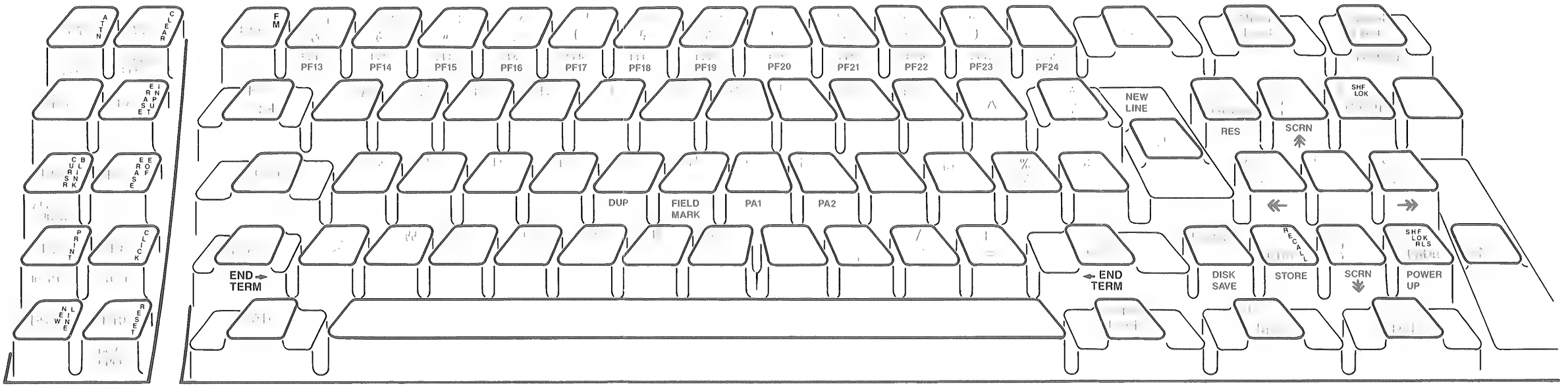
Legends in Print (Black on decals)
These keys are used to invoke special IRMA functions while running the terminal emulator. While in the emulator mode, these keys replace normal PC legends.

Legends in Print (Blue on decals)
These keys are used in combination with the Alternate (Alt) key to perform certain commands or functions. To invoke Alt functions, press the Alt key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PFI function, press simultaneously the Alt and 1 keys.

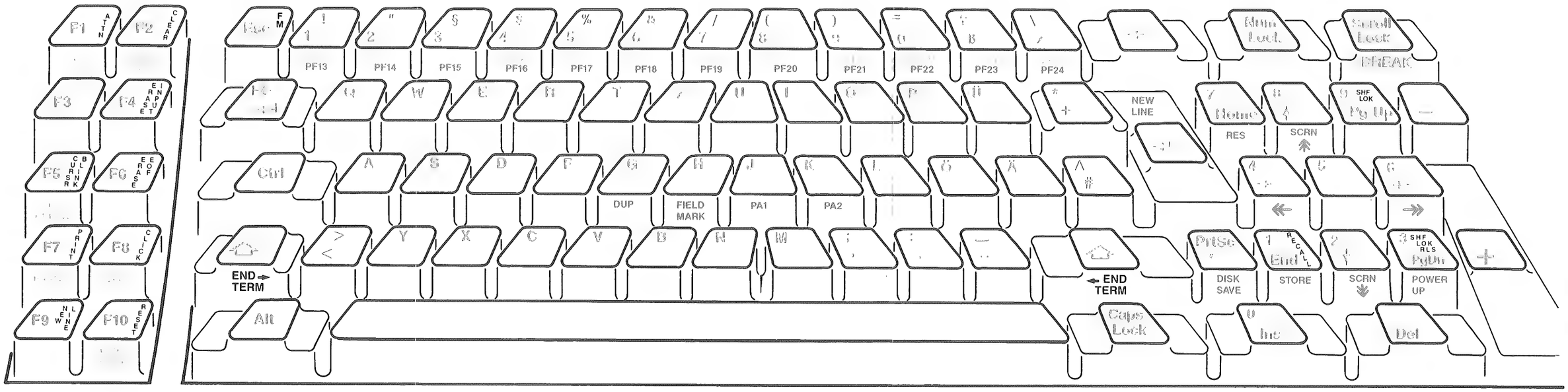
Legends in Print (Red on decals)
These keys are used in combination with the Control (Ctrl) key to perform certain commands or functions. To invoke Ctrl functions, press the Ctrl key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PF13 function, press simultaneously the Ctrl and 1 keys.



Finnish Keyboard



French AZERTY/QWERTY Keyboard



German/Austrian Keyboard

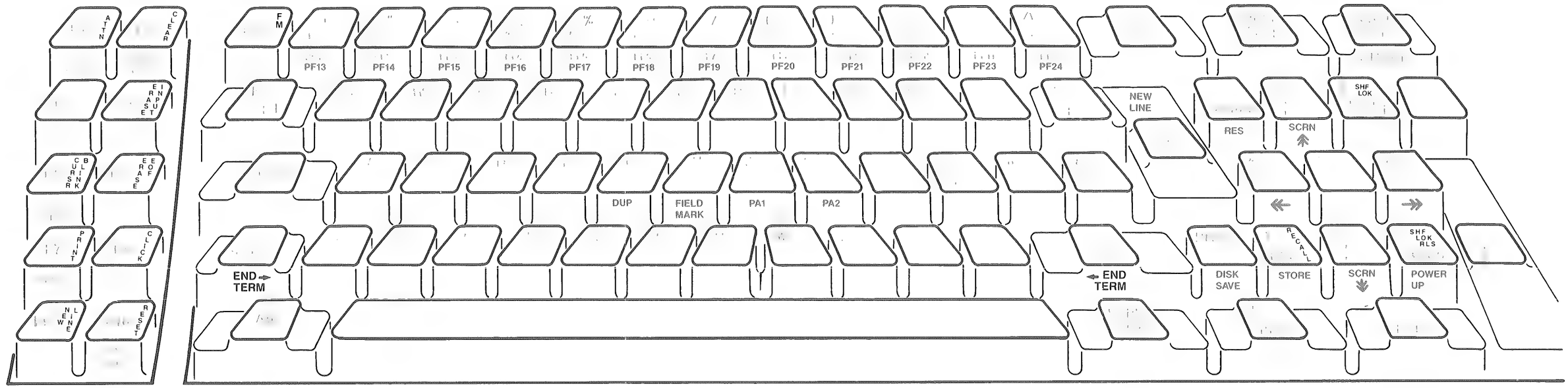
This keyboard chart is provided as a guide to the location of keys that invoke special IRMA functions. The chart may also be used as a guide to placement of the color-coded decals provided in the documentation package.

Legends in Print
These keys represent normal PC functions. Some of these functions may not be available while running the terminal emulator.

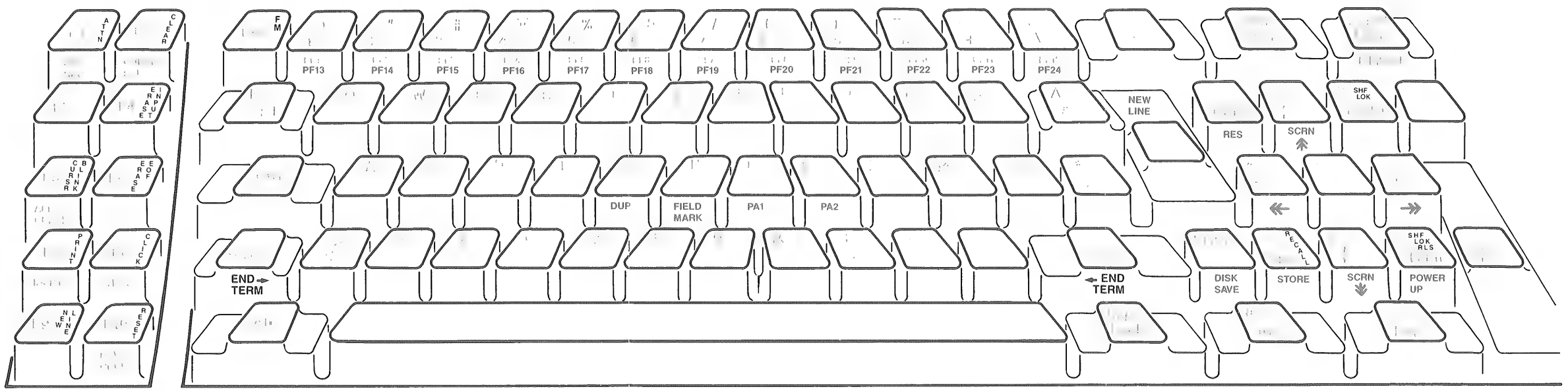
Legends in Print (Black on decals)
These keys are used to invoke special IRMA functions while running the terminal emulator. While in the emulator mode, these keys replace normal PC legends.

Legends in Print (Blue on decals)
These keys are used in combination with the Alternate (Alt) key to perform certain commands or functions. To invoke Alt functions, press the Alt key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PFI function, press simultaneously the Alt and 1 keys.

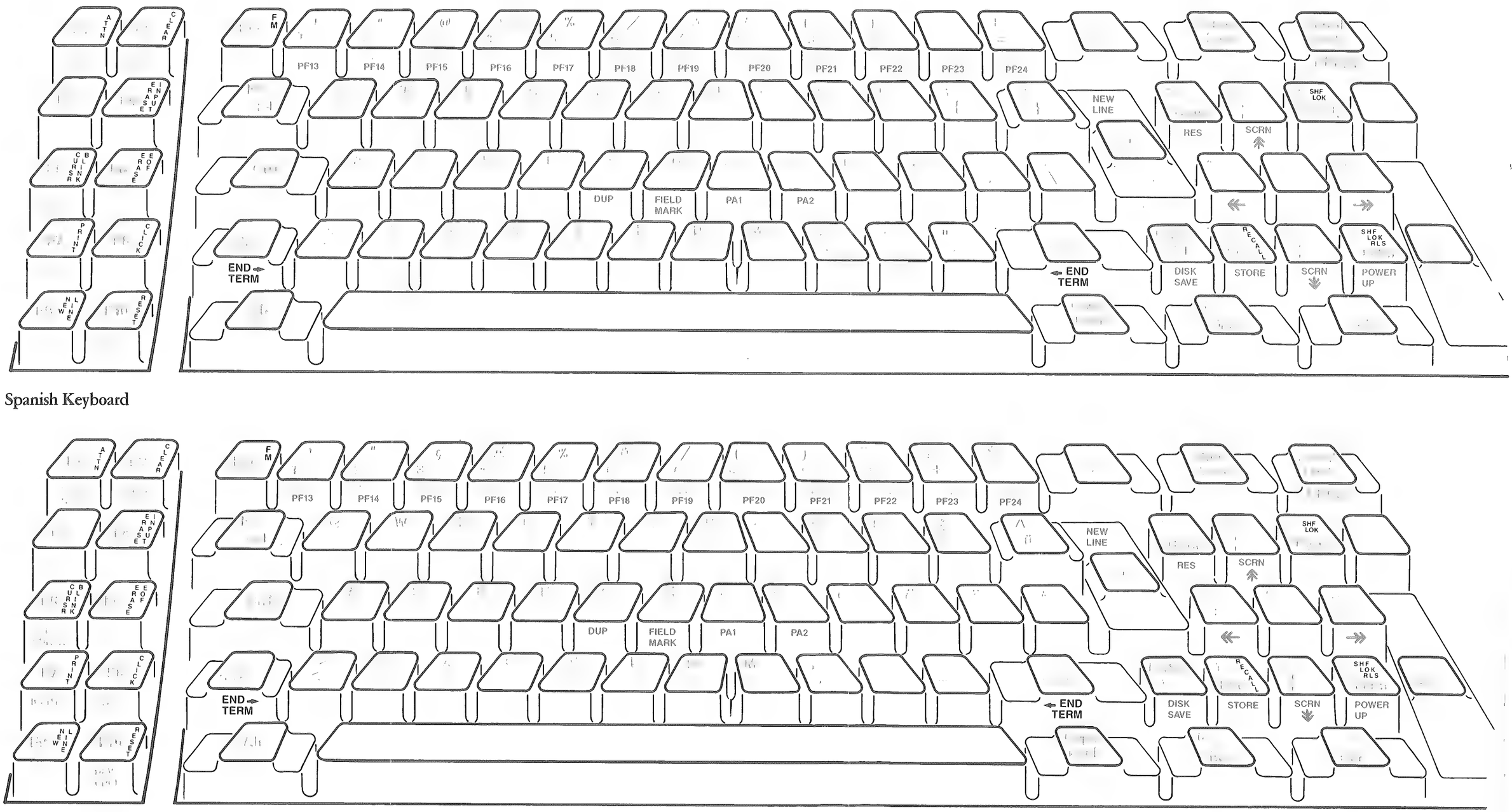
Legends in Print (Red on decals)
These keys are used in combination with the Control (Ctrl) key to perform certain commands or functions. To invoke Ctrl functions, press the Ctrl key while at the same time pressing the key appropriate for the desired function or command. For example, to invoke the PF13 function, press simultaneously the Ctrl and 1 keys.



Italian Keyboard



Norwegian Keyboard



Spanish Keyboard

Swedish Keyboard

Make a Backup Copy of IRMA Software

Before using the IRMA software it is important to make a backup copy. Use the backup copy for daily use. *Do not ever modify the original diskette*. If you have a standard two drive system (A and B) follow the instructions below. If you have a single drive system or a hard disk drive (C) refer to the IBM DOS manual and/or the IBM Guide to Operations for your system.

STEP 1:	**Insert IBM PC DOS diskette into drive A**	To boot system
STEP 2:	**Boot DOS by pressing the Ctrl and Alt keys together. Then press the Del key. Release all three keys at the same time.**	To get PC running. The A prompt should be displayed on the monitor.
STEP 3:	**Enter the DISKCOPY command**	The DOS diskette must still be in drive A when this command is entered.
STEP 4:	**Follow the prompts that are displayed on the screen. (Remove the DOS diskette and place the IRMA diskette into drive A).**	
STEP 5:	**Insert a blank diskette into drive B**	This becomes the target diskette
STEP 6:	**Remove both diskettes when the copy is complete**	Store the original in a safe place. Use the backup copy; *Do not perform GENX routine on the original.*

For more detailed instructions on the DISKCOPY command refer to the IBM PC DOS Manual.

Why You Use the Backup Copy

You should always use a backup copy. If the original should be damaged or altered in any way, you would have no immediate means of recovering the data on the original. As you become more familiar with the options provided in the GENX routine, you may find that you want more than one version available for different applications. It is recommended to customize only unmodified versions of the IRMA emulator software. Consequently, you may need to make additional backups of the original as the need arises. If you modified the original, you would have no means of making additional customized versions.

Special Options You May Need

Before continuing to Chapter 3, "Using IRMA and the Terminal Emulator," you may need to be familiar with some details of a program called GENX. GENX is a program that allows you to, through the use of a menu, make certain changes to the E78 emulator program. These changes may satisfy special requirements demanded by your particular system needs. For instance, GENX allows you to select a keyboard type other than the typewriter-style, or to select from a number of terminal screen types and sizes. You may also use GENX to select certain color features.

If you plan to use IRMA and your PC as a 3278 terminal with a standard typewriter-style keyboard (Mod 2, 80 column by 25 line display), you probably do not need to use the GENX program. If you are using IRMA with a PC-compatible COMPAQ or EAGLE 1600 machine, you MUST use the GENX program to accommodate certain modifications required by these units.

If you're not using a PC-compatible and have no reason to use GENX at this time, your installation is complete. Proceed to Chapter 3, "Using IRMA and the Terminal Emulator," for instructions on how to use IRMA. If you are using a PC-compatible, proceed to the next section 'Additional Steps Required for PC-Compatibles'. If you need further information concerning GENX, refer to Appendix B, "Using GENX."

Additional Steps Required for PC-Compatibles

As mentioned previously, you may not need to have a customized version of IRMA's emulator software. However, if you are using a PC-compatible machine you must use GENX to make some adjustments. The instructions that follow present a step-by-step guide for making these adjustments. If you require additional GENX options, refer to Appendix B for a detailed explanation of GENX.

STEP 1:	**Insert the backup diskette into drive A**	To make IRMA software available
STEP 2:	**Enter E78GEN**	To modify IRMA's emulator software

The following menu appears:

E78 Terminal Emulator Customization Menu

1 — Disable 26th line status display
3 — Make emulator auto-resident
5 — Make 2 color mode default
6 — Make 7 color mode default
8 — Make SHOW COLUMNS default
10 — Convert to pre-1.20 keyboard
12 = Select KEYBOARD & SCREEN type
13 = Set LIGHT PEN correction
14 = PC look-alike patches
15 = Setup COM1: input parameters
17 — Disable NON-DISPLAY display
18 — Force dark blue to cyan
20 — Select IRMAlette operation

99 = Exit GENX program

Your selection:

STEP 3:	**Enter 14**	To select PC look-alike patches for COMPAQ or EAGLE 1600

At this point a submenu is displayed. Its screen appears as follows:

```
PC Look-alike Patches

  1 — Eagle 1600 screen cleanup
  3 — Force Horizontal update sync
  5 — Disallow 26 line color display
  7 — Set cursor position

98 — Return to previous menu        99 — Exit GENX program

Your selection:
```

STEP 4: **Enter the appropriate choices.** To eliminate non-IBM PC machine problems

Item # 1 Eliminates dots that sometimes appear on the CRT. This selection also enables 8086 type screen accesses and should be selected when IRMA/E78 is used in any 8086 based machine.

Item # 3 When selected removes or reduces screen flicker on some non-IBM machines, notably the EAGLE PC.

Item # 5 Should be selected if using the COMPAQ color/monochrome display.

Item # 7 Sets the cursor to the bottom of the character block. Some monochrome displays position the cursor in the center of the character block, rather than at the bottom. This option should be used at the user's discretion whenever cursor requires repositioning.

The other options presented in the GENX main menu may be used to satisfy special requirements demanded by your system. For instance, GENX allows you to select a keyboard type other than the typewriter style such as an APL keyboard or a Mod 3 keyboard. You may also use GENX to make the E78 program, in its customized form, auto-resident or to select certain color features. If you want to examine these options more closely, refer to Appendix B, "Using GENX," for a complete description of GENX.

If no other modifications to the E78 software are required for normal operations, proceed to STEP 5. If you want to make other changes go to STEP 6. To begin using E78 you must exit the GENX routine.

STEP 5: **Enter 99**	To return to DOS operating system.
STEP 6: **Enter 98**	To return to GENX main menu

Your modifications are saved in a file called CE78. When you are ready to access the 3270 host, you should enter CE78 in place of E78 to activate the emulator mode.

You are now ready to begin using IRMA and the E78 software. Chapter 3, "Using IRMA," presents all of the user information required to operate as a 3278 or 3279 terminal from your personal computer.

Chapter 3

Using IRMA and the Terminal Emulator

Introduction

The terminal emulator program is very important to your IRMA unit. This program, referred to as E78, works with IRMA to greatly enhance the capabilities of your IBM PC. With IRMA and its emulator program, the PC now becomes a much more versatile business tool, one that is able to serve you in two ways. Besides serving as a stand-alone personal computer with its word processing and other capabilities, the PC with IRMA becomes your link to the full computing power and data base of the host computer. Both PC and mainframe programs are now available to you.

With the terminal emulator program active, your PC screen has the same features as those found on any of the following IBM terminal models:

- 3278 Mod 2 (monochrome with 24 lines, 80 characters each).
- 3278 Mod 3 (monochrome with 32 lines, 80 characters each).
- 3278 Mod 4 (monochrome with 43 lines, 80 characters each).

If you use a color monitor with your PC, E78 also allows emulation of the IBM Model 3279 character terminal with full seven-color support.

How IRMA Works with its Emulator Program

IRMA actually acts as a translator, translating each keystroke twice. When you press a key on your PC keyboard, IRMA translates that key character or function into a form that can be understood by your IBM mainframe system. Likewise, when information is returned by the IBM mainframe to your screen, IRMA makes sure that the information is in a form that can be understood by your PC.

Customarily, the information sent from the IBM mainframe and displayed on your screen is organized into areas called fields. Key functions that involve the printing of information, or storing or transmitting it, are all organized into this field format.

The characters that define the type of data to be entered in a particular field are called attribute characters. IRMA supports the use of these characters, which define such things as the start of a field, whether information may be entered into the field, whether the field may be detected by a light pen, and others. Other attribute characters called Extended Attribute Characters define character types (blink, reverse video, etc.), character color, or character sets. If you wish to know more about these attribute and extended attribute characters, see Chapter 5, "Technical Reference."

During the installation you should have placed the decals onto the appropriate keys. If you have not done so yet; do so now. The instructions and color coded keyboard charts are found in Chapter 2, "Installation."

E78 allows your PC to function as a 3278 or 3279 terminal. Before beginning the program, make sure that you are completely and properly installed, and that all cable connections are correct. To be certain, ask yourself the following:

- Is your IRMA card properly installed in the PC?
- Do you have a coaxial cable firmly connected, one end to the IRMA port on the rear panel of the PC, and the other end to your IBM 3274 or 3276 controller?
- Is the IRMA coaxial port configured for a 3278/79-type typewriter keyboard?
- Is your PC unit plugged in?

If you answered yes to all of these questions, read on to the next paragraph. If you answered no to any question, turn back to the installation instructions. Follow all installation steps carefully. When you have completed all installation steps, turn again to the questions above.

The IRMA E78 diskette is not a bootable diskette. In order to execute any of the programs contained on this diskette, IBM PC-DOS must be actively running on the PC. The procedures for booting DOS are found in Chapter 2, "Installation," and in the IBM PC-DOS manual.

Also, if you have not already done so, make duplicate copies of your IRMA diskette now. For daily use, use a duplicate copy—not the original. You may also copy the contents of the IRMA diskette on to a hard disk, if your PC is so equipped.

Instructions for making backup copies of a diskette are found in Chapter 2, "Installation," and in the IBM PC-DOS manual.

To begin using E78, follow these steps:

STEP 1:	**Insert the duplicate diskette into the disk drive (unless you have moved the IRMA diskette contents to hard disk).**	To prepare for use
STEP 2:	**Enter: E78 after the prompt**	To activate the basic E78 program
	or	
	E78 <filename>	To activate the E78 program with Screen Save capabilities.
	or	
	E78/n	To activate the E78 program with Screen Store and Recall capabilities
	or	
	E78/n <filename>	To activate the E78 program with Screen Save and Screen Store and Recall capabilities

NOTE: For those users who have modified E78, your modifications are saved in a file called CE78 (Customized E78). You should enter the appropriate disk drive and CE78 to activate the emulator mode. From this point on in this chapter, any reference to E78 should be considered a reference to CE78 as well.

As indicated in STEP 2, there are four ways to access E78. If you do not need to save any screens to diskette, or store screens in memory for later recall, simply enter E78 after the system prompt. Note that if the E78 diskette is not in the working disk drive, you must also enter the disk drive in which the E78 program is located. (For example, if drive A is the working drive, and E78 is located in drive B, enter B:E78 after the system prompt or if E78 has been moved to a hard disk drive, enter C:E78.)

For additional information on "Screen Save" and "Screen Store and Recall," refer to the topic, 'Special Key Functions For Screen Save and Recall,' found later in this chapter.

When IRMA is acknowledged by your IBM mainframe system, the screen exhibits the same system information displayed by a 3278/79 terminal. The first screen displayed is usually the IBM System Logon Banner. At this point, you may use the PC exactly like any 3278/79 terminal. You may access system programs, create new files, or perform any other function of the IBM 3270 network that your MIS department allows you to use. Please be aware that to use the mainframe you must have a valid password. Passwords are normally assigned by the MIS or mainframe systems personnel.

Emulator Mode/PC Mode

If you are using your PC as a 3278/79 terminal, you are said to be in the emulator mode. When you are using your PC for its capabilities alone, you are said to be in the PC stand-alone mode. You may switch back and forth between the 3278/79 emulator mode and the PC stand-alone mode. To exit the emulator mode, simply press both SHIFT keys simultaneously. To re-enter the emulator mode, type E78 in response to the PC prompt.

You have available to you an even more convenient method for moving between the 3278/79 and PC modes. While in the emulator mode, press the CONTROL (CTRL) and HOME keys simultaneously. This key combination makes the emulator program "resident" (resident emulation requires at least 96K of PC memory). When the emulator program is resident, you need not enter E78 each time you wish to re-enter the emulator. Once your emulator program is resident, both exit and re-entry of the emulator is accomplished by simultaneously pressing both SHIFT keys.

STEP 1:	**If you are not in emulator mode, enter the access command appropriate for your application, such as E78 or E78/n <filename>.**	To activate emulator

STEP 2:	**Press CTRL & HOME simultaneously Press ESC to complete sequence.**	To make the E78 "resident"
STEP 3:	**Press both SHIFT keys simultaneously**	To exit Emulator mode

Now do spread sheets or whatever you desire with your PC as a stand-alone machine (just don't re-boot), then

STEP 4:	**Press both SHIFT keys simultaneously**	To re-enter Emulator mode

One limitation exists when using the resident emulator. PC-DOS does not allow you to save screens from a resident program. To compensate for this limitation, you may simultaneously activate both the resident and non-resident forms of the emulator program. To do this, return to the PC stand-alone mode and enter E78 <filename>. This gives you access to a non-resident E78 program. For more information on screen save, refer to the topic 'Special Key Functions for Screen Saves and Recall,' found later in this chapter.

Commands And Functions Available with E78

The following table contains commands and key combination functions most commonly used by the general user. This table is followed by another that lists a number of special commands and functions typically used by the programmer—not the general user. In both tables, the command or key combination is shown in upper case letters and/or numerals on the left. In parentheses underneath the key combinations is a brief explanation of the function for quick identification. The right-hand column of the table contains a more comprehensive explanation of how the command or key combination is used.

Table 3-1 Commands and Key Functions for General User

Command or Key Combination	Function or Use
E78	Activates the terminal emulator program.
E78/n	Activates the terminal emulator with Screen Store and Recall capabilities.
E78 <filename>	Activates the terminal emulator with Screen Save capabilites.
E78/n <filename>	Activates the terminal emulator with both the Screen Store and Recall and Screen Save capabilities.
For each of the following key functions, both keys are pressed simultaneously.	
SHIFT & SHIFT (Exit/enter Emulator)	Exits either non-resident or resident emulator. May also be used to re-enter the resident emulator.
CTRL & HOME (Make resident)	Creates a "resident" copy of the emulator program. (Must have 96K memory.)
CTRL & PgDn (Reset system)	Causes system to reset, as if just powered up. Equivalent function on the 3278 terminal is test switch located to right of the display screen.
CTRL & 4 (Cursor left two positions)	<< — moves cursor left two character positions. (THE '4' IS LOCATED ON THE NUMERIC KEY PAD.) This function is also provided as CTRL & D.
CTRL & 6 (Cursor right two positions)	— >> moves cursor right two character positions. (THE '6' IS LOCATED ON THE NUMERIC KEY PAD.) This function is also provided as CTRL & F.
CTRL & F1 (Color select)	Assigns attribute color code for display (provided color display adapter is use with the PC). If CTRL & F1 are entered, monochrome application programs are displayed in 3279 color.
CTRL & F3 (Show unprotected)	Places dots in unprotected null fields. Useful to check the length of data entry fields.
SHIFT & PrtSc (Print screen)	Prints current screen on local printer (This function can be used by both resident and non-resident emulators.)

The following table shows special key combinations and functions typically used by a programmer. These functions may be of little use to the general user.

Table 3-2 Key Functions for Programming Use

Key Combination	Function or Use
CTRL & F2	Attributes (display buffer codes (0C0H – 0FFH) are displayed as ASCII characters 040H – 07FH. For Example, Attribute 0C0H is displayed as "@". The following chart shows the displayed characters for each attribute.

Conversion of Attribute characters to Display symbols:

Attr.	Char	Attr.	Char	Attr.	Char	Attr.	Char
0C0H	@	0D0H	P	0E0H	"	0F0H	p
0C1H	A	0D1H	Q	0E1H	a	0F1H	q
0C2H	B	0D2H	R	0E2H	b	0F2H	r
0C3H	C	0D3H	S	0E3H	c	0F3H	s
0C4H	D	0D4H	T	0E4H	d	0F4H	t
0C5H	E	0D5H	U	0E5H	e	0F5H	u
0C6H	F	0D6H	V	0E6H	f	0F6H	v
0C7H	G	0D7H	W	0E7H	g	0F7H	w
0C8H	H	0D8H	X	0E8H	h	0F8H	x
0C9H	I	0D9H	Y	0E9H	i	0F9H	y
0CAH	J	0DAH	Z	0EAH	j	0FAH	z
0CBH	K	0DBH	[	0EBH	k	0FBH	{
0CCH	L	0DCH	\	0ECH	l	0FCH	|
0CDH	M	0DDH	]	0EDH	m	0FDH	}
0CEH	N	0DEH	^	0EEH	n	0FEH	~
0CFH	O	0DFH	_	0EFH	o	0FFH	<

For additional information on attributes refer to Chapter 5, 'Technical Reference.'

CTRL & F3 (Show unprotected)	Places dots in unprotected null fields. Useful to check the length of data entry fields.

Special Key Functions For Screen Save and Recall

The Special Key Functions for Screen Save and Recall are commonly used by all levels of IRMA users. These key functions provide a simple means of saving screens to disk and for storing screens for later recall. They were not included in the above Tables because they require that you access E78 with the optional arguments. (The access sequences were mentioned briefly during the initial instructions for activating E78.)

Control & Print Screen

This function allows you to save screens to diskette. This function is only available when you are using the non-resident emulator. However, you may have both a resident and non-resident emulator available. If you are using the resident emulator, you simply need to exit and access the non-resident emulator with the specified filename to use this function. When the CTRL and Print Screen (PrtSc) keys are pressed simultaneously, the screen that is currently being displayed is copied to a file on the diskette. When you first access E78, you must specify the filename in which to store these screens. Each screen that you save is appended to the end of the file. For a complete view of the procedures involved, refer to the following steps:

STEP 1:	**If you have not specified a filename, exit emulator mode by pressing both SHIFT keys.**	To enter PC mode
STEP 2:	**Enter:** **E78 <filename>**	To access non-resident emulator with specified filename in which to save screens. You may also specify a disk drive such as B:filename.
STEP 3:	**Watch screen; the same screen that was displayed when when you exited the emulator mode is redisplayed. Locate the screen to be saved.**	To prepare for the screen save

STEP 4:	**When you have the desired screen displayed, press CTRL and PrtSc simultaneously.**	To save the screen
STEP 5:	**Repeat STEP 4 for each screen to be saved.**	To save additional screens

Once you have specified a filename, you can use that filename repeatedly; however, you may want to sort screens according to content or application. In that event, you may access E78 with one filename, save certain screens, exit 3270 mode, and access E78 with a different filename, and save a different type of screen to that file. Even if you have specified a filename at some earlier point, you must specify that same filename or a new filename each time you access E78 for the purpose of saving screens. If you do not specify a filename when you access E78, you will not be able to save screens to diskette.

The number of screens saved is limited only by the amount of memory you have available on the diskette. It should be noted that to have both a resident and non-resident emulator active at the same time, your PC should be equipped with at least 96K of memory.

Control & End and End

The Control & End key function allows you to save up to nine screens in memory for later recall. This function is available with either the resident or non-resident emulator. The screens are saved in screen buffers, not on the diskette. This function is used in conjunction with the End key function which recalls a screen once it is saved. In order to use these functions, you must access E78 with a "/n" qualifier, where n is the number of screens you wish to have available. You do not need to specify a filename for this function. Follow the steps given below for the procedures required for this function. Assuming that you have not already specified a value for n:

STEP 1:	**Exit 3270 mode by pressing both SHIFT keys.**	To return to PC mode
STEP 2:	**Enter: E78/n (n can be a number from 1 to 9)**	To enable screen store

STEP 3:	**Locate screens you want to store. (You can be in either resident or non-resident mode to use this function.)**	To prepare for store
STEP 4:	**Press both CTRL and END simultaneously.**	To store screen
STEP 5:	**Enter a value from 1-9 after the prompt, "Screen Memory for STORE."**	To assign a value to the 'Screen Memory' that can be used for later recall
STEP 6:	**Repeat STEPS 4 and 5 for each screen to be saved.**	To store additional screens

Once you have stored the screens, the END key is used to recall them. There is no special access to E78 required to use this function and it is available from either the resident or non-resident version of the emulator. As long as you have not rebooted the PC since you stored the screens, they are available for recall at any time.

Pressing the End key causes a prompt to be displayed on the status line, 'SELECT SCREEN MEMORY FOR RECALL.' Press a number key from 1 to 9 to recall a stored screen. Press 0 to return to the current 3270 screen. When you select a number from 1 to 9, that particular screen image is displayed on the screen. To re-display the screen that was present before the recall, enter 0. The following steps explain this procedure:

STEP 1:	**If you are not in 3270 mode, access either the resident or non-resident emulator.**	To access 3270 mode
STEP 2:	**Press END.**	To begin recalling screens
STEP 3:	**Respond to prompt in the the status line by entering the number of the screen to be recalled.**	To select specific screen
STEP 4:	**Repeat STEPS 2 and 3 for each screen to be recalled.**	To recall remaining screens

STEP 5: **Press END.**	To recall screen
STEP 6: **Enter 0 after the prompt on status line.**	To recall the 3270 screen that was displayed before you began recalling the stored screens

While a 'recalled' screen is being displayed, pressing ENTER will also return you to the original 3270 screen that was displayed before the recall. Other keys may also achieve the same result; however, if it is a printable character it will be inserted into the original screen as it is redisplayed. DCA recommends that you use the END function and enter 0 after the prompt on the status line. This avoids inserting unwanted characters into the 3270 display.

When a screen image is displayed, the message 'Screen memory n' is displayed at the right-most end of the status line. This message indicates that the screen image 'n' (1 thru 9) is displayed. There are three error conditions that could occur. Messages appear briefly on the status line to indicate these error conditions. They are:

- Memory not allocated
- Memory empty
- Invalid number (not 1-9)

Memory not allocated means that you have tried to store a screen without specifing the /n value. Memory empty means you have tried to recall a screen without storing one. Invalid number means you have entered a number outside the valid range of 1-9.

Each screen image allocated to memory requires 4K bytes (4096) of system memory. This is in addition to the memory used by the resident emulator. It is recommended that at least 128K of memory be installed in the PC to accommodate full usage of this feature.

Display mode function keys, such as CTRL & F1 or CTRL & F3 (Color select or show attributes), may be used when viewing recalled screens in all modes available to the normal screen. You must be aware that saving a screen with non-displayed fields present allows that screen to be later recalled with that

non-displayed field displayed. A particular caution is appropriate when that non-displayed field contains a password that would be displayed when the screen is later recalled. You are cautioned against STOREing screens containing privileged or secure information in non-displaying fields.

Since you can access the store and recall functions from either the resident or non-resident emulators, you can also recall any of the screens stored from either emulator, regardless of the emulator in use at the time of the store. For example, you have saved nine screens throughout your 3270 session. Some of them were stored during a non-resident session and some were stored during a resident session. Now you would like to recall the screens and save them to a disk file. Remember that to save screens to a disk file, you must be using the non-resident emulator. Exit whatever emulator you are using; re-access E78 with E78 <filename>. You can now recall the nine stored screens and save them to diskette.

Remember earlier in this chapter there were four ways to access E78 discussed. If you plan on storing screens and/or saving screens to diskette, it would save you some time if you initially accessed E78 with both the specifications of n and filename. That way you would not have to exit 3270 mode and re-access E78. Refer back to STEP 2 under the topic, 'How to Use the Terminal Emulator.'

The following table summarizes the above key functions.

Table 3-3 Screen Save and Recall Functions

Key Combination	Function
CTRL & Prt Sc (Saves screens to disk)	Copies screens to disk. You must be using the non-resident emulator to use this function. You must also specify a filename when you access E78.
CTRL & END (Screen Store)	Stores up to 9 screens in memory. You can use this function with either the resident or non-resident versions of E78. You must specify the /n value when accessing E78.
END (Screen recall)	Recalls screens that have been stored using the CTRL & END Function.

Simply recalling stored screens does not require accessing E78 with a special qualifier (n or filename). However, if you want to save those recalled screens to diskette you do need to specify a filename when you access E78. If the PC has been rebooted between the time you stored the screens and the time you try to recall them, the memory in which the screen were stored is empty. The screens are not permanently stored. As long as the PC is not rebooted the screen remain in memory and can be recalled at any time.

System Message Characters

All IBM 3278/79 terminals reserve one line as a status line for display of system messages such as error codes. The PC does not display some characters required for system message symbols. In these cases, the PC displays an alternate PC character in place of the 3278 character. The following chart shows 3278 system message symbols, their corresponding PC symbol, and the message represented by the symbols.

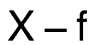
These symbols are typically used in combination. Examples of these combinations are illustrated in the next section, 'Typical Combination Symbols.'

3278-2	PC	Message
P	P	Program
S	S	Used with X— to indicate symbol keyed is not available
A	a	
▲	⌂	Insert mode
B	b	
6	6	Online to a 3276 controller
►	►	Used in combination with other symbols

3278-2	PC	Message
		Used in combination with other symbols
		Used in combination with other symbols
		Used to indicate "not working"
		Shift lock
		Operator
		SNA attachment
		Alpha lock
		Unowned session
		Operator's program
		Used with X and nn to indicate communication link error
		Used in combination with other symbols.
		Used in combination to form — + z —
		Used in combination to form other symbols.
		Left half of clock
		Right half of clock
		Input inhibited
		Used in combination to form other symbols
		Used in combination to form other symbols
		Right portion of Printer failure message
		Left half of security key

3278-2	PC	Message
		Right half of security key
		Online to a 3274 controller
		BSC or Non-SNA attachment
		Symbol for card
		System operator

Typical Combination Symbols

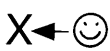
 Function Unavailable

 Security key off

 Printer not working

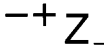
 Printer busy

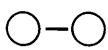
 Operator unauthorized for specified printer

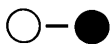
 Go elsewhere, action has been attempted which is invalid for field

 Operator entered invalid number in field

 Message received from system operator and rejected

 Communication link producing errors

 Printer assignment

 Printer active

Chapter 4

File Transfer

Introduction

Today's business man and woman have greater communication needs than ever before. Access to the full computing power and database of the mainframe computer is now a necessity, and possessing the necessary terminal capabilities is a major first step towards this goal. But this mainframe access is not complete without the ability to transfer information in either direction between your PC and the host computer. This ability is called *file transfer.*

With this capability, you are able, as a personal computer user, to transfer the data required for your PC applications directly to and from the mainframe.

File transfer offers you a number of very attractive advantages. For instance, you are able to gain access to the information you require without relying on such time-consuming manual data transfer methods as reporting and data entry. In addition, since file transfer can relieve the mainframe computer of word processing, executive decision-making applications, such as spread sheets, and other applications more appropriately performed on the personal computer, you and others benefit from a more efficient use of all data communications systems.

DCA provides as part of the IRMA package, two file transfer utilities. IRMA*link FT78T*, the transfer facility for TSO environments and IRMA*link FT78X*, the transfer facility for CMS environments.

Features

- Provides file transfer under both CMS and TSO IBM mainframe operating environments.
- Supports both mainframe to micro (download) and micro to mainframe (upload) transfer.
- Supports transfers either to or from the IBM mainframe.
- For simple file transfer, provides a HELP function and question-and-answer format.

Files

The IRMA file transfer programs supplied on your file transfer diskette are compatible with both Rev. 1.1 and 2.0 of DOS, all revisions of E78 software, and firmware revisions 3.14 and above. The following five files are supplied on your program diskette. The executable file transfer program and PC78TABS.OVR must reside in the default disk drive. If they do not, the file transfer will not work.

FT78X.EXE	Executable file transfer program for CMS XEDIT
FT78T.EXE	Executable file transfer program for TSO
PC78.XED	IRMA XEDIT profile for use with the CMS file transfer utility. Instructions for use of this profile are provided later in this section.
PC78TABS.OVR	Must be on file transfer diskette
FTSAMPLE.TXT	Sample text file for testing file transfer

Environment

The host environment is limited to VM/CMS, using XEDIT, or MVS/TSO using the EDIT function of TSO. The PC must have at least 128K of memory. If you are using the VM/CMS transfer environment, your instructions begin on the next page. MVS/TSO user instructions follow the CMS section.

General Information

As an IRMA*link FT78X* user, you need to have some familiarity with the CMS XEDIT editor. If you do not have this knowledge, see your MIS or other technical support personnel before attempting any file transfer procedures.

The IRMA*link FT78X* program may be used in either of two formats. The first format uses a question-and-answer dialogue to provide information for the transfer. DCA recommends the use of this question-and-answer format for all users except those who have already become very familiar with IRMA CMS file transfer.

The second format places all information in one command line. This format also allows you to add "switches" to the command line for verification and confirmation of information used in the file transfer. This format is also useful with batch files.

IRMA*link FT78X* file transfer allows the transfer of binary files only from PC to PC. It does not support the transfer of host binary files to PC's.

When the file is sent to the host, the binary data is translated into an intermediate text file format. This text file is stored in a special format which is not directly usable on the host. When the file is transmitted back to a PC, it is reconstructed as PC binary data.

This binary mode is implemented for such operations as the transfer of .BAS type files from PC to PC. To successfully transfer binary data, the IRMA XEDIT profile must include the option of lower-case characters, or the host filetype must allow lower case by default.

The IRMA file transfer diskette contains a help file that lists commands for both transfer formats, as well as switch values important to the single line format. For more information concerning this help file, see the topic called 'IRMA*link FT78X* Help File.'

ASCII TO EBCDIC Conversion

EBCDIC conversion occurs when a file is sent from the PC to the host. The following ASCII characters are, in all cases, translated by PC78TABS.OVR into the equivalent EBCDIC characters.

ASCII	EBCDIC
[	¢
]	|
^	¬

CMS / XEDIT Operation Losses

Due to the use of POWERINPUT in XEDIT to improve performance on a SEND, two ASCII characters are not available: " ^" and "[" are reserved as special characters. The " ^"and "[" characters are effectively changed to spaces and are NOT recoverable on subsequent receives. If either character is encountered during a send operation, a warning is displayed on the CRT.

IRMA*line FT78X* setup procedures are described below.

Setup Procedures For IRMAlink FT78X

Before using CMS file transfer, you must perform the following setup procedures:

STEP 1:	**Access the E78 program**	To gain access to the mainframe
STEP 2:	**Log on to the mainframe and access the CMS XEDIT mode.**	To be able to enter XEDIT profile
STEP 3:	**Enter the CMS XEDIT profile. (The profile is listed after the sample of the HELP screen)**	To prepare the mainframe for transfer
STEP 4:	**Exit 3270 mode by pressing both SHIFT keys**	To return to PC mode

At this point, it would be useful to display the IRMA*link FT78X* help screen. This is an optional step.

STEP 5: **Enter FT78X/H** To display the HELP screen.

When you enter FT78X/H, the PC displays the following information:

FT78X/H — Help File

The file transfer utility may be used in either of two formats:

```
FT78X
  or
FT78X {switches} <local_name>
<host_name> <host_type>
```

The first format provides information through a question-and-answer dialogue. The second format contains all information in the command line and is useful with batch files. If the second format includes fewer than the required number of items, you are prompted for the remaining items.

Global Switches

/B — Binary transfer mode

/C — Confirm the pending operation prior to execution

/F — Display host file size on receive

/O — Override the delete-file query and automatically delete the local file if it exists

/R — Receive a file from the host

/S — Send a file to the host

/V — Display data on CON: during the transfer

CMS Profile

The PC78.XEDIT profile, sets up the functions used by IRMA*link FT78X* version of IRMA file transfer. The profile must be entered at the host with the IRMA filename and XEDIT filetype. The profile must be entered and saved once for each USERID using the file transfer programs. The profile may be

entered in lower or upper case, but it must otherwise be entered *EXACTLY AS SHOWN*. Enter the following profile:

```
SET SCALE OFF
SET NUMBER ON
SET CURLINE ON 3
SET CMDLINE BOTTOM
SET NULLS ON
SET PF6 QQUIT
SET PF18 QQUIT
SET PF7 FORWARD
SET PF19 FORWARD
SET PF8 TOP
SET PF20 TOP
SET PF9 FILE
SET PF21 FILE
```

The following optional item allows lower case as the default (must be set for binary transfer), and may be included in the profile if desired.

```
SET CASE MIXED
```

When you have entered the profile, save it as PC78.XEDIT before exiting 3270 emulator mode. Turn to the following pages for instructions concerning the question-and-answer transfer format.

Using IRMAlink FT78X In Question-And-Answer Dialogue Mode

If you have not already completed the procedures for setup, do so at this time. To transfer files under CMS after setup is accomplished, perform the following steps:

STEP 1:	**Enter 3270 mode. Access E78 or press both SHIFT keys if emulator is resident.**	To access CMS

STEP 2:	**Log on to mainframe and access CMS XEDIT mode.**	To transfer data, you must have both CMS XEDIT available on one end of the transfer and the PC file on the other
STEP 3:	**Exit 3270 mode by pressing both SHIFT keys**	To return to PC mode
STEP 4:	**Enter FT78X. This command can be entered only in PC mode.**	To activate the question and answer format

The following series of questions are displayed when you enter FT78X. Answer the questions using the explanations provided below.

PROMPT: Confirm selections prior to transfer? [Y,N]

ENTER: Y for Yes, N for No

This selection causes a line similar to the following text line to appear on the screen at the end of the question and answer prompts, but before the actual transfer takes place. Answering Yes causes a listing of the file received and the save file:

Receive from host <filename>, save as local <filename>.

This confirmation is followed by:

OK to continue? [Y/N]

If the filenames are listed correctly, enter Y for Yes. If filenames are not correct, enter N for No, and the IRMA*link FT78X* program is terminated. Restart the IRMA*link FT78X* program at Step 1 above to re-specify the operation.

PROMPT: Transfer direction. [R/S]

R = Receive a file on the PC from the host

S = Send a file from the PC to the host

ENTER: R or S

PROMPT: Transfer binary file. [Y/N]

ENTER: Y or N

If a binary data file is to be transferred, enter Y for Yes.

IRMA binary file transfer allows the transfer of binary files from PC to PC only. It does not support the transfer of host binary files to PC's.

PROMPT: Display copy to CON: [Y/N]

ENTER: Y or N

If you want to view the data on the screen as it is transferred, enter Y for Yes. To avoid interruptions in the transfer, do not alternate between the PC and 3278 modes while the transfer is taking place.

For the transfer to take place, the program must know the source and destination files. Whether sending or receiving a file, prompts for supplying information are in the same order. You must always supply the PC filename, called the local filename, and the host filename and filetype. The IRMA*link FT78X* program interprets which file is source and which is destination.

PROMPT: What is the local filename?

Must be one word and an acceptable PC-DOS name. It may include a drive specifier.

ENTER: <filename>

PROMPT: Host filename

This filename must match your host system naming conventions, except that the name should not include the data set type extension.

ENTER: <filename>

PROMPT: What is the filetype on the host?

Because it allows variable length records of up to 255 characters, XEDIT is the recommended filetype. However, you may also use DATA (80 characters) or SCRIPT (132 characters).

To transmit a file from the host to the PC, the file must already exist at the host. If the PC is receiving the file and the filename already exists, the following question is displayed:

PROMPT: Do you want to delete and write over that file? [Y/N]

ENTER: Y or N

If Y is entered, the file is replaced with the file from the mainframe.

If N is entered, you are prompted by FT78X to re-specify the filename.

If the host filename is the destination (on send) and is non-empty, a clear-data option is offered. Unlike the local file, which can be deleted, the actual host file is not deleted — only the contents of the file.

PROMPT: Host filename already exists, clear it? [Y/N]

ENTER: Y or N

Y clears the file and allows the user to continue.

N terminates the file transfer program.

If you have specified the confirm option, the text line specifying the receiving and sending files is displayed on the screen. At this point, you may continue or terminate the file transfer program.

The transfer is complete when the A, B, or C prompt appears. This prompt is your only indication that the transfer is complete. When you have received the prompt, you may continue. If you have completed all activity with the host, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

How To Use The Single Command Line Format

If you have not already completed the CMS setup procedures, do so at this time. To transfer files under CMS after setup is accomplished, perform the following steps:

STEP 1:	**Enter 3270 emulator mode. (Enter E78 or if you have a resident version, press both SHIFT keys.)**	To begin transfer session
STEP 2:	**Log on to the mainframe and access the CMS XEDIT mode.**	You must be in XEDIT mode before transfer can be attempted
STEP 3:	**Exit 3270 emulator mode.**	To return to PC mode. The file transfer utility must be initiated in PC mode.
STEP 4:	**Enter (in one line)**	To activate transfer

FT78X{switches} <local filename>
<host_filename> <host_filetype>

See the list below for acceptable switch values which must also be entered on the same line as the command. If you fail to include all of the necessary switches, you will be prompted for those omitted.

Global Switches

/B — Binary transfer mode

/C — Confirm the pending operation prior to execution

/F — Display host file size on receive

/O — Override the delete-file query and automatically delete the local file if it exists

/R — Receive a file from the host

/S — Send a file to the host

/V — Display data on CON: during the transfer

The transfer is complete when the A, B, or C prompt appears on the screen. This prompt is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no other use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

IRMAlink FT78T

As an IRMA*link FT78T* user, you need to have some familiarity with the TSO EDITOR. If you do not have this knowledge, see your MIS or other technical support personnel before attempting any file transfer procedures.

IRMA*link FT78T* may be used in either of two formats. The first format uses a question-and-answer dialogue to provide information for the transfer. DCA recommends this question-and-answer format for all users except those who have become very familiar with IRMA*link FT78T*.

The second format places all information in one command line. This format also allows you to add "switches" to the command line for verification and confirmation of information used in the file transfer. The single line format is also useful with batch files.

IRMA file transfer allows the transfer of host binary files only from PC to PC; it does not support the transfer of host binary files to PC's.

When the file is sent to the host, the binary data is translated into an intermediate text file format. This text file is stored in a special format which is not directly usable on the host. When the file is transmitted back to a PC, it is reconstructed as PC binary data.

This binary mode is implemented for such operations as the transfer of .BAS type files from PC to PC. To successfully transfer PC binary data, the IRMA XEDIT Profile must include the option of lower case characters, or the host filetype must allow lower case by default.

Under TSO, files usually contain line numbers; however, files on the PC do not have line numbers. Files sent to the host from the PC are automatically numbered, beginning at 10 and incremented by 10, for a maximum 9,999 lines of text. Files received by the PC are temporarily renumbered, beginning at 1 and incremented by 1, for a maximum 99,999 lines. Before the file is written to the diskette on the PC, these line numbers are automatically removed.

In addition, by including the NONUM operand, un-numbered data sets can be sent and received. This causes IRMA*link FT78T* to ignore line numbers entirely and not attempt to renumber the data set on a receive.

The TSO version of file transfer allows the user to fully specify the data set name and associated EDIT operands. The data set name may contain no imbedded spaces but may contain an extension (e.g. FTSAMPLE.TXT).

FT SAMPLE.TXT, for example, would be an incorrect name format, due to its imbedded space. The name may be as detailed as necessary, allowing easy specification of partitioned data sets. The operands may also be as detailed or as simple as necessary and are not limited to a single word. Thus, users may add items such as ASIS (lower case) and/or NONUM (un-numbered data set). Several examples are included later in this discussion.

The IRMA file transfer diskette contains a "help" file that lists commands for both formats, as well as switches used with the single line format. For more information concerning this help file, see the section called 'TSO Help File.'

ASCII To EBCDIC Conversion

EBCDIC conversion occurs when a file is sent from the PC to the host. The following ASCII characters are, in all cases, translated by PC78TABS.OVR into equivalent EBCDIC characters.

ASCII	EBCDIC
[	¢
]	|
^	¬

Operational Losses

The EDIT mode of TSO changes the following characters to a ":" during transfer:

\	}
`	~
{	

Even though E78 echoes the ASCII symbol, this results in a non-recoverable loss of information.

For example, the following represents a table on the IBM PC, as sent to the mainframe using TSO/EDIT:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_
`	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	{	¦	}	~	~

Below is the table as it was received on the PC back from the mainframe:

0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	!	"	#	$	%	&	'	(	)	*	+	,	-	.	/
0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?
@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
P	Q	R	S	T	U	V	W	X	Y	Z	[	:	]	^	_
:	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o
p	q	r	s	t	u	v	w	x	y	z	:	¦	:	:	

IRMA*link FT78T* setup procedures begin on the following page.

Before using TSO file transfer, you must perform the following procedures:

STEP 1:	**Access E78 by pressing both SHIFT keys or entering E78.**	To gain access to the mainframe
STEP 2:	**Log on to the mainframe and access the TSO EDITOR. The TSO READY prompt is displayed. (The TSO file transfer cannot be performed under the SPF EDITOR.)**	To key in the TSO profile
STEP 3:	**Enter the TSO Profile. The profile is listed after the sample of the IRMA*link FT78T* Help screen.**	To prepare the mainframe for file transfer
STEP 4:	**Exit 3270 mode by pressing both SHIFT keys.**	To return to PC mode

At this point it would be useful to display the HELP screen for a listing of the two formats and switches that can be used with the single command line format.

STEP 5:	**Enter FT78T/H.**	To display HELP screen

The screen displays the following information:

FT78X/H — Help File

The file transfer utility may be used in either of two formats:

```
FT78X
  or
FT78X {switches} <local_name>
<host_name> <operands>
```

The first format provides information through a question-and-answer dialogue. The second format contains all information in the command line and is useful with batch files. If the second format includes fewer than the required number of items, you are prompted for the remaining items.

Global Switches

/B — Binary transfer mode

/C — Confirm the pending operation prior to execution

/O — Override the delete-file query and automatically delete the local file if it exists

/R — Receive a file from the host

/S — Send a file to the host

/V — Display data on CON: during the transfer

TSO Profile

The IRMA file transfer programs require that certain statistics be set one time under the TSO EDITOR for each transfer user. These statistics are available under a command called "PROFILE." You may view the profile from the mainframe READY prompt. To display the profile, enter the word PROFILE, and press ENTER or RETURN. The following profile should be displayed.

```
CHAR(0) *
LINE(0) *
PROMPT *
INTERCOM *
NOPAUSE *
NOMSGID *
NOMODE *
NOWTPMSG *
NORECOVER *
PREFIX [xxxx]
DEFAULT LINE/CHARACTER DELETE CHARACTERS IN
   EFFECT FOR THIS TERMINAL *
```

The profile should contain all of the above parameters. It may include others as required by your system. All parameters followed by an asterisk should be included in the profile just as they are listed here. When the profile is displayed, it is displayed in three lines, listing the parameters from left to right. If you need to change or add any of these parameters to your profile, refer to the next section, called 'How To Set or Change Your TSO Profile.' Set your profile to match the one shown above. When your profile is correct and is successfully entered, save the profile.

At this point, you have completed TSO file transfer setup procedures. To transfer files, follow the instructions provided in 'Using TSO File Transfer In Question-And-Answer Dialogue Mode' located later in this chapter.

How To Set Or Change Your TSO Profile

To set or change any of the statistics within your TSO profile, enter:

PROFILE <statistic>

where <statistic> is the value to which you wish the profile statistic to be changed. For example, if your present profile statistic is NOPROMPT and you wish to change the statistic to PROMPT, enter:

PROFILE PROMPT

If the profile change is successful, the READY prompt is displayed on your screen. This prompt is your only indication of a successful change. If the profile setting or change is in error, the following message is displayed:

INVALID KEYWORD, <statistic>
REENTER

where <statistic> is the profile statistic in error.

To display the profile after a change, enter the word PROFILE, and press ENTER or RETURN.

Using IRMAlink FT78T In Question-And-Answer Dialogue Mode

To transfer files under TSO after setup is accomplished, perform the following steps:

STEP 1:	**Enter 3270 mode by pressing both SHIFT keys if emulator is resident or by entering E78.**	
STEP 2:	**Log on to Mainframe and access TSO EDITOR. The TSO READY prompt is then displayed.**	To transfer data you must have the TSO EDITOR available
STEP 3:	**Exit 3270 mode by pressing both SHIFT keys.**	To return to PC mode

STEP 4: **Enter FT78T. This command must be entered from PC mode.** To set up the transfer

The following series of questions are displayed. Answer the questions using the explanations provided below.

PROMPT: Confirm selections prior to transfer? [Y,N]

ENTER: Y for Yes, N for No

This selection causes a line similar to the following text line to appear on the screen at the end of the question and answer prompts, but before the actual transfer takes place. Answering Yes causes a listing of the file to be received and the file to be saved.

Receive from host <filename>, save as local <filename>.

This confirmation is followed by:

OK to continue? [Y/N]

If the filenames are listed correctly, enter Y for Yes. If filenames are not correct, enter N for No, and the IRMA*link FT78T* program is terminated. To re-specify the operation, restart the IRMA*link FT78T* program.

PROMPT: Transfer direction. [R/S]

R = Receive a file on the PC from the host
S = Send a file from the PC to the host

ENTER: R or S

PROMPT: Transfer binary file. [Y/N]

ENTER: Y or N

If a binary data file is to be transferred, enter Y for Yes.

IRMA binary file transfer allows the transfer of binary files from PC to PC only. It does not support the transfer of host binary files to PCs.

PROMPT: Display copy to CON: [Y/N]

ENTER: Y for Yes if the transferred data is to be displayed on the screen.

To avoid interruptions in the transfer, do not alternate between the PC and 3278 modes while the transfer is taking place.

For the transfer to take place, the program must know the source and destination files. Whether sending or receiving a file, prompts for supplying information are in the same order. You must always supply the PC filename, called the local filename, and the host filename and filetype. The IRMA*link FT78T* program interprets which file is source and which is destination.

PROMPT: What is the local filename?

Must be one word and an acceptable PC-DOS name. It may include a drive specifier.

ENTER: <filename>

PROMPT: What is the Data-set-name?

ENTER: <data-set-name>

This name must include a valid dataset type extension.

PROMPT: What are the operands for host?

ENTER: <operands> [none]

"[none]" is the default. If no operands are required, simply enter a <NL> or <ENTER>.

To transmit a file from the host to the PC, the file must already exist at the host. The destination filename should not exist prior to the transfer. However, if the destination filename does exist and is non-empty, a clear-data option is offered after all PROMPTS have been answered. If the PC is receiving the file and the filename already exists, the following question is displayed:

PROMPT: Do you want to delete and write over that file? [Y/N]

ENTER: Y or N

If Y is entered, the file replaced with the file from the mainframe.

If N is entered, you are prompted by FT78T to re-specify the filename.

If the host filename is the destination (on send) and is non-empty, a clear-data option is offered. Unlike the local file, which can be deleted, the actual host file is not deleted—only the contents of the file.

PROMPT: Host file already exists, clear it? [Y/N]

ENTER: Y or N

Y clears the file and allows the user to continue.

N terminates the file transfer program.

If you have specified the confirm option, the text line specifying the receiving and sending files is displayed on the screen. At this point, you may continue or terminate the file transfer program.

The transfer is complete when the A, B, or C prompt appears on the screen. This prompt is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

How To Use The IRMAlink FT78T Single Command Line Format

To transfer files under TSO after setup is accomplished, perform the following steps:

STEP 1:	**Enter 3270 emulator mode. (Enter E78 or if you have a resident version, press both SHIFT keys.)**	To begin transfer session
STEP 2:	**Log on to the mainframe and access the TSO EDITOR mode.**	You must be in EDITOR mode before transfer can be attempted
STEP 3:	**Exit 3270 emulator mode.**	To return to PC mode. The file transfer utility must be initiated in PC mode.
STEP 4:	**Enter (in one line)**	To activate transfer

FT78T{switches} <local filename>
<host_filename> <host_filetype>

(See the following list for acceptable switches.)

Global Switches

/B — Binary transfer mode

/C — Confirm the pending operation

/O — Override the delete-file querry and automatically delete the local file if it exists on the receive. At the host on send, deletes the only the contents of file, not actual file.

/R — Receive a file from the host.

/S — Send a file to the host

/V — Display data on CON: during the transfer.

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample One: FTSAMPLE.TXT

A copy of the sample test file, FTSAMPLE.TXT, is to be sent to the host as the numbered, sequential data set, TEST.TXT. The following dialogue achieves the transfer:

PROMPT	RESPONSE
A>	FT78T <RETURN>
Confirm selections prior to transfer [Y/N]	N <RETURN>
Transfer direction [R/S]	S <RETURN>
Transfer binary file [Y/N]	N <RETURN>
Display copy to CON:	N <RETURN>
Local filename:	FTSAMPLE.TXT <RETURN>
Data set name:	TEST.TEXT <RETURN>
Operands: [none]:	<RETURN>

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample Two: Partition Transfer

A copy of the member, MEM1, in the partitioned, numbered data set PPS.DATA, is to be stored in the IBM PC as FILE1.DAT. The following dialogue achieves the transfer.

PROMPT	RESPONSE
A>	FT78T <RETURN>
Confirm selections prior to transfer [Y/N]	N <RETURN>
Transfer direction [R/S]	R <RETURN>
Transfer binary file [Y/N]	N <RETURN>
Display copy to CON:	N <RETURN>
Local filename:	FILE1.DAT <RETURN>
Data-set-name:	1. PPS.DATA(MEM1) <RETURN> or 2. PPS(MEM1) <RETURN>
Operands: [none]:	1. <RETURN> or 2. DATA <RETURN>

When entering the data set name, the first entry includes the filetype and, therefore, does not require the filetype to be specified as an operand. If using the second entry, the filetype must be entered as an operand. When entering the data set name and operands for this example, match the numbered responses for each entry.

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample Three: Binary Transfer

The BASICA program, PROG.BAS, is to be sent to the host as TEMPPROG.DATA. Since the binary transfer mode is to be used, lower case is required on the host. The following dialogue achieves the transfer.

PROMPT	RESPONSE
A>	FT78T <RETURN>
Confirm selections prior to transfer [Y/N]	N <RETURN>
Transfer direction [R/S]	S <RETURN>
Transfer binary file [Y/N]	Y <RETURN>
Display copy to CON:	N <RETURN>
Local filename:	PROG.BAS <RETURN>
Data set name:	TEMPPROG.DATA <RETURN>
Operands: [none]:	ASIS <RETURN>

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample Four: Single Command Line

To perform the same transfer described in the previous example, this time using the single command line format, enter the following data:

FT78T/S/B PROG.BAS TEMPPROG.DATA ASIS

or

FT78T/S/B PROG.BAS TEMPPROG DATA ASIS

Note that in the first case, DATA is specified as part of the data set name, while in the second case, DATA is an operand.

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample Five: PC-To-PC Transfer

Using the binary transfer mode, first upload the specified file to the mainframe. When the file is located on the mainframe in its storage state, other PC's, using the same file transfer utility, may access the file and download it to a PC. The file is sent to the PC in its original state (.BAS or WORDSTAR™ format).

The transfer is complete when the A, B, or C prompt appears on the screen. This is your only indication that the transfer is complete. When you receive this prompt, you can continue with whatever activity you want. If you have no further use for the host 3270, return to 3270 mode by pressing both shift keys or entering E78, and log off the host.

Sample Six: To Specified Disk Drive

To specify an alternate disk drive for the file coming from the mainframe, precede the destination filename with the device name, such as B:filename or C:filename.

Chapter 5

Technical Reference

Introduction

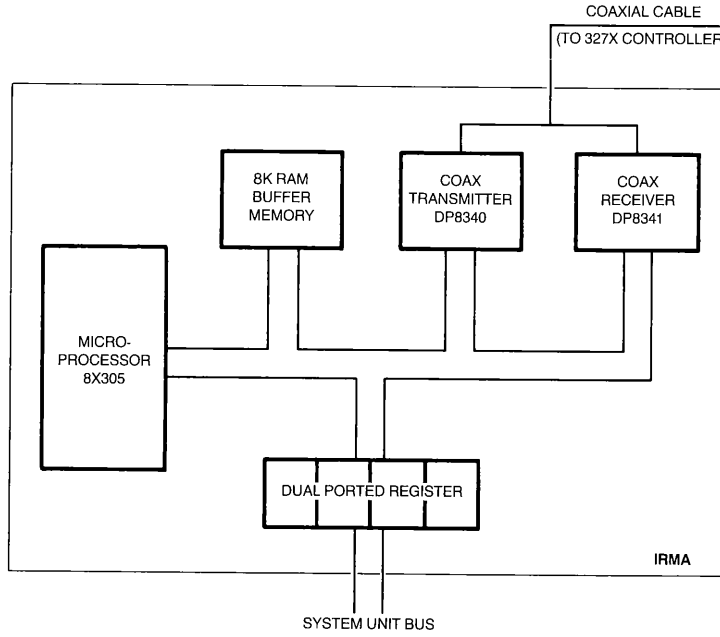
IRMA is a printed circuit board which plugs into the IBM PC System Unit. The board (card) can be installed in any slot in the PC and provides a back panel BNC connector for attachment by coaxial cable to either a 3274, 3276, or integral controller.

IRMA operates in a stand-alone mode, using an on-board microprocessor to handle the 3270 protocol and screen buffer. Whenever power is applied to IRMA, it responds to commands from the controller as if an IBM 3278 terminal were attached to the coaxial cable. The IRMA screen buffer is accessed from the PC System Unit as an I/O device, the device codes being 220H to 227H. IRMA does not occupy any of the memory address space.

To meet the requirements of the 3270 protocol, the Decision Support Interface (DSI) uses high-speed microprocessor technology that is independent of the 8088 microprocessor of the System Unit. This allows the user to ignore the timing requirements of 3270, and operate with a buffer of data just as the 3278 screen does.

When operating, the DSI takes commands from a four-byte dual-ported register array in addresses 220H to 223H. This array is accessed by I/O from the System Unit. The four single byte words are arranged as the command and up to three arguments. Command words allow the System Unit program to read or write bytes in the screen buffer, send keystrokes, and access the special features available on the DSI.

This array is also handled on the DSI by the microprocessor which manages the 3270 protocol. When the DSI is idle between messages from the controller, any commands left in the array by System Unit programs are processed as required. This processing occurs only when the higher priority 3270 communication is idle. This idle state is indicated by a busy/done flag mechanism for both the DSI and System Unit microprocessors. This allows the System Unit to declare that a new command is available to the DSI, and for the DSI to signal the completion of this command.



8X305 Microprocessor

The 8X305 microprocessor provides the intelligence required for 3270 protocol. Polling and answering, data transfer, handshaking, and screen buffer maintenance are performed by this processor.

DP8340 and DP8341 3270 Coax Transmitter/Receiver Interface

These two IC's provide interface from the microprocessor to the 3270 coaxial cable. Serialization and deserialization of data take place in these two parts.

Screen Buffer

The DSI contains 8K bytes of fast RAM memory for screen buffers and temporary storage. These 8K bytes represent 3K bytes each for the screen and Extended Attribute buffers, leaving 2K bytes for local storage used by the 8X305 processor.

Dual-Port Register Array

The four-byte dual-ported register array is shared by the 8X305 and System Unit 8088 processors. These registers are used for all communication between the two microprocessors. Data to be transferred from one processor to the other is written into specific locations (addresses) in the array (220H-223H), and the "Command Request" flag (226H) is set. When the receiving processor has read the register, this flag is cleared. Each processor can test the state of this flag to see if data transfer can begin, and to determine when the transfer is complete.

The dual-ported communication array is used to pass commands from the System Unit and its program to the DSI. The registers are organized as four I/O addresses (220H to 223H), using one eight-bit word at each address. The base address of the standard DSI command is 220H. Arguments for the command are placed into addresses 221H, 222H, and 223H. Standard command operation for the DSI requires the user program to set values into the dual port register as appropriate for each command. First, the user program sets the "Command Request" flag. IRMA then reads the data, performs the operation, and leaves any resulting data in the dual port array. The Command Request flag is cleared at that time.

Sixteen commands are defined for the DSI program. The following table lists these command codes and their values.

Code	Command Definition
0	Read buffer data
1	Write buffer data
2	Read status/cursor position
3	Clear main status bits
4	Send keystroke
5	Light pen transmit
6	Execute Power-on-Reset
7	Load trigger data and mask
8	Load trigger address
9	Load attention mask
10	Set terminal type
11	Reserved—Do not use
12	Read terminal information
13	No-op
14	Return revision ID and OEM number
15	Reserved—Do not use

The argument structures related to each command code are detailed in the "Command Descriptions" section.

Programming Considerations

The DSI is accessed as an I/O device on the System Unit bus. It uses eight I/O device codes, 220H through 227H. Device codes 220H, 221H, 222H, and 223H are a dual- access register array attached to both processors. The four bytes are used to pass commands from the 8088 to the DSI and to return any answers. Device codes 224H and 225H are reserved for future use. Device codes 226H and 227H are used for Command Request and Attention Request flags, respectively.

All commands use Word 0, device code 220H, as a command selection register. To begin a command, the program must set Word 0 equal to the command number. The other three words are used to pass the arguments of the command. When the specified command is completed, Word 0 contains the main status bits, and the other three words contain output data.

When a command is sent to the mainframe by a user or by an automated PC process, the terminal is placed into a waiting state until the command is completed. During this waiting period, the system places an "X" on the status line at row 0, column 8 of the screen buffer. This "X" remains on the status line until the command is completed. Before any new command is sent, the "X" must have disappeared. Note that the system may rewrite this "X" position several times before the command is actually completed. During this time, the "X" may flicker and appear at times to be cleared, while not actually being so.

If you are sending keystrokes through a user program instead of directly from the keyboard, your program should do the following to check for the X clock.

1. Clear Buffer Dirty in Main Status
2. Check Buffer Dirty for 100 ms to 200 ms
3. If Buffer is not clear, go back to step 1
4. If Buffer stays clear, check X position
5. If X present, wait until clear
6. When position has been clear for 100 ms, Clear Main Status bits
7. Send next command

Programmer's Notes on Status Bits

Main Status bits indicate specific conditions. The "Aux Status Change" bit is set anytime the Aux Status changes. The "Trigger Occurred" bit is set whenever the trigger data match occurs (see Load Trigger Data and Mask command (7), "Command Description" section). The "Key Buffer Empty" bit is set when the key scan code buffer is empty (see Send Keystroke command(4), 'Command Descriptions' section). The "Unit Reset" bit is set whenever the controller sends a 3270 reset command. The "Buffer Modified" bit is set when any buffer write occurs. The "Cursor Position Set" bit is set whenever the controller positions the terminal's cursor in the buffer.

The "Aux Status" bits are defined as follows: The "Unit Polled" bit is set by a poll command. This bit clears after reading. Since the controller polls about 1,000 times per second, this status bit is set often. The other 6 Aux status bits are defined exactly as 3270 protocol defines them.

The "Command Interrupt Request" bit should not be used to tell when another command can be started. The hardware flag "Command Request" must be used for this purpose. The "Command Interrupt Request" bit clears when a command is begun, and is set at the end of the command. However, it is not cleared immediately upon "Command Request".

Since the "Key Buffer Empty" bit does not clear immediately upon the "Keystroke" command, the empty bit, used to check the buffer before sending the keystroke command, is guaranteed valid only if no command is in progress. Again, use the hardware flag to check for a command in progress before checking the empty flag bit.

The Main Status in Word 0 is updated each time any of the conditions specified occurs. This word can be read and used with the above limitations at any time. The "Read Status" command is not necessary except to read the "Aux Status" or "Cursor Position". The returned Main Status byte consists of 8-bit flags as follows:

Bit	Meaning
7 (MSB)	Aux Status change has occurred (*)
6	Trigger Occurred (*)
5	Key Buffer Empty
4	Fatal IRMA hardware error
3	Unit Reset by controller (*)
2	Command interrupt request (+)
1	Buffer Modified (*)
0	Cursor Position Set, or search backward (*)

(*) = Bits which must be cleared by user program

(+) = Bit allows the attention request/interrupt request mechanism to be used with commands. Programmed I/O operation should use the hardware flag for all busy/done checking.

(MSB) = Most Significant Bit

The bit flags in the Aux Status are defined as follows:

Bit	Meaning
7 (MSB)	UNUSED
6	Unit Polled since last Status Re:
5	Sound Alarm
4	Display Inhibited
3	Cursor Inhibited
2	Reverse Cursor Enabled
1	Cursor Blink Enabled
0	Keyboard Click Enabled

Read Buffer Data Command (0)

For a read data command, Word 0 is set to zero. Word 1 is the low order 8 bits of the buffer address to be read. Word 2 is the high order 3 bits (right justified) of the address. Word 3 is unused. Upon completion of the command, Word 2 contains the associated Extended Attribute Data, and Word 3 contains data from the specified buffer location.

Internal IRMA screen buffers correspond to supported terminal types: Mod 2, 2000 characters (25 lines by 80 characters); Mod 3, 2800 (35x80); and Mod 4, 3520 (44x80). In each case, the last line represents the status line. Even though the screen is displayed with the status line at the bottom of the screen, this line is actually the first line in memory. The 'DSI Screen Buffer' section provides in table form the starting addresses for each line of each supported terminal type.

The Read Data command returns the buffer data, corresponding EAB data, and the main status. Each command returns main status in Word 0.

Word	Value	Input	Output
0	0	Command to DSI	Main Status
1	ADDR(L)	Address (low) to read	UNUSED
2	ADDR(H)	Address (high) to read	EAB* Data from DSI
3	DATA	UNUSED	Data from DSI

* EAB — Extended Attribute Buffer (Refer to Attribute Character explanations later in this chapter.

Write Buffer Data Command (1)

The write data command is used to write (modify) the contents of the screen buffer. Like read data, Word 1 and Word 2 contain the address of the buffer location to which data is to be written. Word 3 is the data to be written. At command completion, the buffer is updated and main status returned.

Word	Value	Input	Output
0	1	Command to DSI	Main status
1	ADDR(L)	Address (low) for write	UNUSED
2	ADDR(H)	Address (high) for write	UNUSED
3	DATA	Data for write	UNUSED

Read Status/Cursor Position Command (2)

This command reads the current status and cursor position from the DSI. The status is returned as two bytes of bit flags. The cursor address is in the same format as the buffer address in read/write data commands.

Word	Value	Input	Output
0	2	Command to DSI	Status
1	ADDR(L)	UNUSED	Cursor address (low)
2	ADDR(H)	UNUSED	Cursor address (high)
3	AUX	UNUSED	Aux status

Clear Main Status Bits Command (3)

This command clears one or more of the main status bits.

Word	Value	Input	Output
0	3	Command to DSI	Main status
1	UNUSED	UNUSED	UNUSED
2	UNUSED	UNUSED	UNUSED
3	MASK	Bit clear mask	UNUSED

Five of the main status bits are set by specific conditions, but cleared only upon command. This allows each bit to be tested and cleared by different sections of the program. The clear mask controls bits to be cleared. For each bit set to 1 in the clear mask, the corresponding bit in the status is cleared (i.e. set to 0). The clear mask can be used to reset multiple bits. Note that the returned status reflects the status AFTER the clear command has been executed.

For ease of implementation, Word 0 is always maintained as the most current status. The normal program sequence would include the following: a read and test of Word 0, a branch on condition to a service routine specific to the bits found to be set, and the Clear Main Status Bits command.

Send Keystroke Command (4)

The send keystroke command causes the DSI to send the controller a key scan code. This is the function used to simulate a key active condition. The key scan code is the exact code which a 3278 terminal would normally send, NOT an ASCII or EBCDIC character code.

Word	Value	Input	Output
0	4	Command to DSI	Main status
1	UNUSED	UNUSED	UNUSED
2	UNUSED	UNUSED	UNUSED
3	CODE	Key scan code to send	UNUSED

This command causes the Key Buffer Empty flag in the status byte to clear. It also checks the status of the Command Request flag. The Key Buffer Empty flag is guaranteed to be valid when a command is not in progress. Before the key scan code can be sent, the Key Buffer Empty flag must have a value of one.

Send Selector Pen Location (5)

This command causes the cursor position of the light pen to be sent to the controller. This code is NOT in cursor address format.

Word	Value	Input	Output
0	5	Command to DSI	Main Status
1	ROW	Row on screen	UNUSED
2	FIELD ID	Field ID on screen	UNUSED
3	UNUSED	UNUSED	UNUSED

Execute Power-On-Reset Command (6)

This command causes the DSI to appear to the controller as if the terminal has just been reset. The command signals the controller that the DSI needs power-up service and initialization.

Word	Value	Input	Output
0	6	Command to DSI	Main status
1	UNUSED	UNUSED	UNUSED
2	UNUSED	UNUSED	UNUSED
3	UNUSED	UNUSED	UNUSED

Load Trigger Data AND Mask Command (7)

This command loads a trigger system with data and mask values. This system places a "watch" on a specific buffer memory location. The watch may be set to check for an exact match on a new value, an inexact (masked) match on a new value, or on any change to the current value. The mask word determines which type of test occurs.

For a trigger to occur, the data and buffer MUST match for each 1 bit in the mask. To make an exact compare, the mask is set to 0FFH (all ones), and the data is set to the desired value.

When a match occurs, the Trigger Occurred bit in the main status is set. If the mask does not contain all ones, only those bits which are 1 are checked for a match. For example:

Buffer Value	0 1 0 1 0 1 0 1
Search Value	0 1 0 1 0 1 1 0
Bit by Bit Compare (1's = Difference) (Exclusive OR)	0 0 0 0 0 0 1 1
Mask	1 1 1 1 1 1 1 0
Logical AND of Mask and Compare	0 0 0 0 0 0 1 0

The result is non-zero, and the trigger does not occur.

The special case of a mask of all zeros is used to handle a test for change of state. At the time the mask is set to 0, the current value of the location in memory is saved by IRMA. This is then compared and any change is reported as a trigger.

Word	Value	Input	Output
0	7	Command to DSI	Main status
1	DATA	Data value for compare	UNUSED
2	MASK	Mask value for compare	UNUSED
3	UNUSED	UNUSED	UNUSED

Load Trigger Address Command (8)

The load trigger address command sets the buffer position for the data/mask compare. This address is in the same format as that of a cursor position and data read/write. After executing the Load Data and Address Trigger commands, the Trigger occurred bit should be cleared. (Clear Main Status Bits Command (3)). The Load Address command should follow the Load Data command.

Word	Value	Input	Output
0	8	Command to DSI	Main status
1	ADDR(L)	Address (low) for checking	UNUSED
2	ADDR(H)	Address (high) for checking	UNUSED
3	UNUSED	UNUSED	UNUSED

Load Attention Mask Command (9)

The attention mask is applied to the regular status word. Whenever a bit is SET in the status word, the corresponding bit in the Attention Mask is checked. If this bit is one, the Attention Request flag is set. This only applies to bits changing from zero to one.

Word	Value	Input	Output
0	9	Command to DSI	Main status
1	UNUSED	UNUSED	UNUSED
2	UNUSED	UNUSED	UNUSED
3	MASK	Mask for status change	UNUSED

Set Terminal Type Command (10)

This command is used to change the response of the interface to the Read Terminal ID command from the controller.

Word	Value	Input	Output
0	10	Command to DSF	Main Status
1	UNUSED	UNUSED	UNUSED
2	UNUSED	UNUSED	UNUSED
3	UNUSED	New Terminal ID (See definition below)	UNUSED

Response Definition

Upper Four Bits:

Keyboard Type	
0	Reserved
1	APL with numeric lock
2	Text with numeric lock
3	Reserved
4	Reserved
5	APL
6	Text
7	Reserved
8	Data Entry 2 with numeric lock
9	Data Entry 1 with numeric lock
A	Typewriter with numeric lock
B	Reserved
C	Data Entry 2
D	Data Entry 1
E	Typewriter
F	No keyboard

Lower Four Bits:

Screen Type		
0000	Reserved	
0010	960	Mod 1
0100	1920	Mod 2
0110	2560	Mod 3
1000	Unknown	
1010	Reserved	
1100	3564	Mod 5
1110	3440	Mod 4

If the terminal type is the same as the current definition then no action is taken, otherwise POR status is issued and the new terminal type stored.

Read Terminal Information Command (12)

Command used to read various terminal conditions.

Word	Value	Input	Output
0	12	Command to DSI	Main status
1	UNUSED	UNUSED	Starting Page Number of EAB (Page = 256 bytes)
2	UNUSED	UNUSED	Starting Page Number of Internal Variables
3	UNUSED	UNUSED	Current Terminal Type

Return Revision ID and OEM Number Command (14)

Unique OEM numbers may be assigned when the proms are programmed.

Word	Value	Input	Output
0	14	Command	Main status
1	UNUSED	UNUSED	Low two digits of revision number in BCD
2	UNUSED	UNUSED	High two digits of Rev. number
3	UNUSED	UNUSED	OEM number— normally 0

Command Request/Attention Request Flags

Two flags allow the 8X305 microprocessor and System Unit 8088 CPU to handshake over commands. The two flags are Command Request and Attention Request. The 8088 can set Command Request, indicating that a new command has been placed in the dual-ported memory. The Attention Request flag is set by the 8X305 processor to indicate a status change with Attention Mask bit set in the corresponding bit. The Attention flag, set by the DSI 8X305 processor can be cleared only by the 8088. The current status of both flags can be read by the 8088 (and the 8X305).

The status flag read command is an I/O read on device code 227H. The resulting eight-bit number contains both flags as follows:

Bit	Meaning
7	Attention Request flag is set by DSI and cleared by user program in the System Unit.
6	Command Request flag is set by the user program in the System Unit to indicate a new command, and is cleared by the DSI after the command is accepted.
5-0	Unused.

To set the Command Request flag, the user program should execute an I/O write to device code 226H. By writing to 226H, bit 6 of 227H is set. To clear the Attention Request flag, the program should execute an I/O write to device code 227H. In either case, the data written is unimportant.

Device Code	Input	Use
226H	I/O Write	Set Command Request Flag
227H	I/O Write	Clear Attention Request Flag
227H	I/O Read	Read Current Flags

Key Scan Codes

In normal 3270 terminal operation, each keystroke is sent as a special key scan code to the controller. The controller responds by updating the screen (or screen buffer) to show the echo of this new keystroke. In the DSI, keystrokes are sent to the controller via a keystroke command using a key scan code. The controller generated screen buffer update occurs just like a terminal, and the program can read this buffer as desired. Note that some characters are generated by multiple scan codes, such as shift up, character, shift down.

Key scan codes are specific to 3270 systems, and are NOT ASCII or EBCDIC. The following tables list the keys and the proper scan codes:

Belgian Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
Q	4D,60,CD	q	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
A	4D,70,CD	a	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
Z	4D,76,CD	z	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
W	4D,79,CD	w	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
#	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
+	4D,26,CD	6	26
\	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Belgian Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	^	11
PF2	4F,22,CF	¨	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	&	15
PF8	4F,28,CF	*	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase EOF	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
à	12	Resert34	
ù	4D,12,CD	Enter	18
é	7E	Space	10
ç	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
'	30	Tab bkwd	35
?	4D,30,CD	Home	4F,35,CF
[	4D,1B,CD	Backspace	31
è	1B	<	09
`	0F	>	4D,09,CD
]	4D,0F,CD	Alt cursr	4F,54,CF
		Ident	4F,36,CF
		TEST	4F,57,CF

Canadian-French Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
)	4D,20,CD	0	20
!	4D,21,CD	1	21
@	4D,22,CD	2	22
#	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
¨	4D,26,CD	6	26
&	4D,27,CD	7	27
*	4D,28,CD	8	28
(	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Canadian-French Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	=	11
PF2	4F,22,CF	+	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	.	32
PF5	4F,25,CF	/	4D,32,CD
FF6	4F,26,CF	à	15
PF7	4F,27,CF	<	4D,15,CD
PF8	4F,28,CF	Attn	50
PF9	4F,29,CF	Sys Req	4F,50,CF
PF10	4F,20,CF	Cursor	51
PF11	4F,30,CF	Clear	4F,51,CF
PF12	4F,11,CF	Erase	4F,53,CF
PF13	40	Blink	54
PF14	41	Erase EOF	55
PF15	42	Print	56
PF16	43	Click	57
PF17	44	Return	08
PF18	45	Up	0E
PF19	46	Down	13
PF20	47	Left	16
PF21	48	Right	1A
PF22	49	Dup	5F
PF23	4A	Mark	5E
PF24	4B	Insert	0C
PA1	4F,5F,CF	Del	0D
PA2	4F,5E,CF	Reset	34
°	12	Enter	18
"	4D,12,CD	Space	10
;	7E	Shift up	CD
:	4D,7E,CD	Alt down	4F
é	14	Alt up	CF
?	4D,14,CD	Tab fwd	36
—	30	Tab bkwd	35
Under-		Home	4F,35,CF
score _	4D,30,CD	Backspace	31
ù	4D,1B,CD	<	09
è	1B	>	4D,09,CD
`	3D	Alt cursr	4F,54,CF
´	4D,3D,CD	Ident	4F,56,CF
		TEST	4F,57,CF

Danish Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
#	4D,23,CD	3	23
¤	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Danish Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	\	11
PF2	4F,22,CF	`	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	^	15
PF8	4F,28,CF	ü	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
ø	12	Reset	34
Ø	4D,12,CD	Enter	18
æ	7E	Space	10
Æ	4D,7E,CD	Shift up	CD
—	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
+	30	Tab bkwd	35
?	4D,30,CD	Home	4F,35,CF
å	4D, 1B, CD	Backspace	31
Å	1B	<	09
¦	0F	>	4D,09,CD
*	4D,0F,CD	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

English (Domestic) Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
)	4D,20,CD	0	20
!	4D,21,CD	1	21
@	4D,22,CD	2	22
#	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
¬	4D,26,CD	6	26
&	4D,27,CD	7	27
*	4D,28,CD	8	28
(	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

English (Domestic) Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	=	11
PF2	4F,22,CF	+	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	.	32
PF5	4F,25,CF	\	15
FF6	4F,26,CF	:	4D,15,CD
PF7	4F,27,CF	Attn	50
PF8	4F,28,CF	Sys Req	4F,50,CF
PF9	4F,29,CF	Cursor	51
PF10	4F,20,CF	Clear	4F,51,CF
PF11	4F,30,CF	Erase	4F,53,CF
PF12	4F,11,CF	Blink	54
PF13	40	Erase EOF	55
PF14	41	Print	56
PF15	42	Click	57
PF16	43	Return	08
PF17	44	Up	0E
PF18	45	Down	13
PF19	46	Left	16
PF20	47	Right	1A
PF21	48	Dup	5F
PF22	49	Mark	5E
PF23	4A	Insert	0C
PF24	4B	Del	0D
PA1	4F,5F,CF	Reset	34
PA2	4F,5E,CF	Enter	18
'	12	Space	10
"	4D,12,CD	Shift up (L)	CD
;	7E	Shift down (L)	4D
:	4D,7E,CD	Shift up (R)	CE
/	14	Shift down (R)	4E
?	4D,14,CD	Alt down	4F
—	30	Alt up	CF
Under-		Tab fwd	36
score _	4D,30,CD	Tab bkwd	35
!	4D, 1B, CD	Home	4F,35,CF
¢	1B	Backspace	31
`	3D	<	09
~	4D,3D,CD	>	4D,09,CD
{	0F	Alt cursr	4F,54,CF
}	4D,0F,CD	Ident	4F,56,CF
		TEST	4F,57,CF

English (UK) Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
#	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
£	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
'	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29
'	4D,3D,CD		
\	3D		

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

English (UK) Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	¬	11
PF2	4F,22,CF	Over-	
PF3	4F,23,CF	score	4D,11,CD
PF4	4F,24,CF	,	33
PF5	4F,25,CF	<	4D,33,CD
FF6	4F,26,CF	.	32
PF7	4F,27,CF	>	4D,32,CD
PF8	4F,28,CF	{	15
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
'	12	Reset	34
*	4D,12,CD	Enter	18
;	7E	Space	10
+	4D,7E,CD	Shift up	CD
/	14	Alt down	4F
?	4D,14,CD	Alt up	CF
—	30	Tab fwd	36
=	4D,30,CD	Tab bkwd	35
`	4D, 1B, CD	Home	4F,35,CF
@	1B	Backspace	31
}	0F	¦	09
		Under-	
		score _	4D,09,CD
		Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

Finnish Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
§	4D,23,CD	3	23
¤	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Finnish Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	é	11
PF2	4F,22,CF	É	4D,11,CD
PF3	4F,23,CF	å	33
PF4	4F,24,CF	Å	4D,33,CD
PF5	4F,25,CF	ä	32
FF6	4F,26,CF	Ä	4D,32,CD
PF7	4F,27,CF	^	15
PF8	4F,28,CF	ü	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
.	12	Reset	34
:	4D,12,CD	Enter	18
,	7E	Space	10
;	4D,7E,CD	Shift up	CD
ö	14	Alt down	4F
Ö	4D,14,CD	Alt up	CF
+	30	Tab fwd	36
?	4D,30,CD	Tab bkwd	35
Under-		Home	4F,35,CF
score _	4D,1B,CD	Backspace	31
–	1B	<	09
'	0F	>	4D,09,CD
*	4D,0F,CD	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

French AZERTY Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
Q	4D,60,CD	q	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
A	4D,70,CD	a	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
Z	4D,76,CD	z	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
W	4D,79,CD	w	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
§	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
+	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

French AZERTY Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	^	11
PF2	4F,22,CF	¨	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	&	15
PF8	4F,28,CF	*	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
ù	12	Reset	34
°	4D,12,CD	Enter	18
é	7E	Space	10
è	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
'	30	Tab bkwd	35
?	4D,30,CD	Home	4F,35,CF
ç	4D, 1B, CD	Backspace	31
à	1B	<	09
`	3D	>	4D,09,CD
£	4D,3D,CD	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

French QWERTY Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
§	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
+	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

French QWERTY Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	^	11
PF2	4F,22,CF	¨	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	*	15
PF8	4F,28,CF	&	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
ù	12	Reset	34
°	4D,12,CD	Enter	18
é	7E	Space	10
è	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
?	30	Tab bkwd	35
'	4D,30,CD	Home	4F,35,CF
ç	4D,1B,CD	Backspace	31
à	1B	<	09
£	4D,3D,CD	>	4D,09,CD
`	3D	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

German/Austrian Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Z	4D,78,CD	z	78
Y	4D,79,CD	y	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
§	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

German/Austrian Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	`	11
PF2	4F,22,CF	´	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	+	15
PF8	4F,28,CF	*	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
ä	12	Reset	34
Ä	4D,12,CD	Enter	18
ö	7E	Space	10
Ö	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
ß	30	Tab bkwd	35
?	4D,30,CD	Home	4F,35,CF
Ü	4D, 1B, CD	Backspace	31
ü	1B	<	09
#	3D	>	4D,09,CD
^	4D,3D,CD	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

Italian Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
£	4D,23,CD	3	23
$	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Italian Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	¦	11
PF2	4F,22,CF	^	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	+	15
PF8	4F,28,CF	*	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
à	12	Reset	34
°	4D,12,CD	Enter	18
ò	7E	Space	10
ç	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
?	30	Tab bkwd	35
'	4D,30,CD	Home	4F,35,CF
é	4D,1B,CD	Backspace	31
è	1B	<	09
ù	3D	>	4D,09,CD
§	4D,3D,CD	Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

Norwegian Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
#	4D,23,CD	3	23
¤	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Norwegian Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	\	11
PF2	4F,22,CF	`	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	^	15
PF8	4F,28,CF	ü	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
æ	12	Reset	34
Æ	4D,12,CD	Enter	18
ø	7E	Space	10
Ø	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-score _	4D,14,CD	Alt up	CF
+	30	Tab fwd	36
?	4D,30,CD	Tab bkwd	35
å	4D, 1B, CD	Home	4F,35,CF
Å	1B	Backspace	31
|	3D	<	09
*	4D,3D,CD	>	4D,09,CD
		Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

Spanish Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
|	4D,21,CD	1	21
"	4D,22,CD	2	22
@	4D,23,CD	3	23
Pts	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Spanish Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	¬	11
PF2	4F,22,CF	¨	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	+	15
PF8	4F,28,CF	*	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
{	12	Reset	34
[	4D,12,CD	Enter	18
ñ	7E	Space	10
Ñ	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-		Alt up	CF
score _	4D,14,CD	Tab fwd	36
´	30	Tab bkwd	35
?	4D,30,CD	Home	4F,35,CF
`	4D, 1B, CD	Backspace	31
\	1B	<	09
		>	4D,09,CD
		Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF
		]	4D,0F,CD
		}	0F

Swedish Key Scan Codes (3270)

Key	Scan Code (HEX)	Key	Scan Code (HEX)
A	4D,60,CD	a	60
B	4D,61,CD	b	61
C	4D,62,CD	c	62
D	4D,63,CD	d	63
E	4D,64,CD	e	64
F	4D,65,CD	f	65
G	4D,66,CD	g	66
H	4D,67,CD	h	67
I	4D,68,CD	i	68
J	4D,69,CD	j	69
K	4D,6A,CD	k	6A
L	4D,6B,CD	l	6B
M	4D,6C,CD	m	6C
N	4D,6D,CD	n	6D
O	4D,6E,CD	o	6E
P	4D,6F,CD	p	6F
Q	4D,70,CD	q	70
R	4D,71,CD	r	71
S	4D,72,CD	s	72
T	4D,73,CD	t	73
U	4D,74,CD	u	74
V	4D,75,CD	v	75
W	4D,76,CD	w	76
X	4D,77,CD	x	77
Y	4D,78,CD	y	78
Z	4D,79,CD	z	79
=	4D,20,CD	0	20
!	4D,21,CD	1	21
"	4D,22,CD	2	22
§	4D,23,CD	3	23
¤	4D,24,CD	4	24
%	4D,25,CD	5	25
&	4D,26,CD	6	26
/	4D,27,CD	7	27
(	4D,28,CD	8	28
)	4D,29,CD	9	29

NOTE: Both SHIFT keys can generate uppercase characters. However, for the purpose of this chart, only 4D and CD were used.

Swedish Key Scan Codes (3270) continued

Key	Scan Code (HEX)	Key	Scan Code (HEX)
PF1	4F,21,CF	é	11
PF2	4F,22,CF	É	4D,11,CD
PF3	4F,23,CF	,	33
PF4	4F,24,CF	;	4D,33,CD
PF5	4F,25,CF	.	32
FF6	4F,26,CF	:	4D,32,CD
PF7	4F,27,CF	^	15
PF8	4F,28,CF	ü	4D,15,CD
PF9	4F,29,CF	Attn	50
PF10	4F,20,CF	Sys Req	4F,50,CF
PF11	4F,30,CF	Cursor	51
PF12	4F,11,CF	Clear	4F,51,CF
PF13	40	Erase	4F,53,CF
PF14	41	Blink	54
PF15	42	Erase EOF	55
PF16	43	Print	56
PF17	44	Click	57
PF18	45	Return	08
PF19	46	Up	0E
PF20	47	Down	13
PF21	48	Left	16
PF22	49	Right	1A
PF23	4A	Dup	5F
PF24	4B	Mark	5E
PA1	4F,5F,CF	Insert	0C
PA2	4F,5E,CF	Del	0D
ä	12	Reset	34
Ä	4D,12,CD	Enter	18
ö	7E	Space	10
Ö	4D,7E,CD	Shift up	CD
–	14	Alt down	4F
Under-score _	4D,14,CD	Alt up	CF
+	30	Tab fwd	36
?	4D,30,CD	Tab bkwd	35
å	4D,1B,CD	Home	4F,35,CF
Å	1B	Backspace	31
'	3D	<	09
*	4D,3D,CD	>	4D,09,CD
		Alt cursr	4F,54,CF
		Ident	4F,56,CF
		TEST	4F,57,CF

The screen buffer maintained in the DSI can be read using the Read Data command. The user program must supply the address in the screen buffer as part of the Read Data command, and receive the data at that location upon completion.

The buffer contains the number of characters normally displayed on the screen of the particular terminal type (Mod 2, 2000; Mod 3, 2800; Mod 4, 3520). The first line of the screen begins in buffer location 50H. Each successive line then consists of 50H (80 decimal) characters in consecutive order. The following tables list starting addresses for each line of Mod 2 and Mod 3 terminal types.

MOD 2 Terminal Type

Line #	Starting Hex	Address Decimal
1	50	80
2	A0	160
3	F0	240
4	140	320
5	190	400
6	1E0	480
7	230	560
8	280	640
9	2D0	720
10	320	800
11	370	880
12	3C0	960
13	410	1040
14	460	1120
15	4B0	1200
16	500	1280
17	550	1360
18	5A0	1440
19	5F0	1520
20	640	1600
21	690	1680
22	6E0	1760
23	730	1840
24	780	1920
Status (Mod 2)	0	0

MOD 3 Terminal Type

Line #	Starting Hex	Address Decimal
25	7D0	2000
26	820	2080
27	870	2160
28	8C0	2240
29	910	2320
30	960	2400
31	9B0	2480
32	A00	2560
Status (Mod 3)	0	0

MOD 4 Terminal Type

Line #	Starting Hex	Starting Decimal
33	A50	2640
34	AA0	2720
35	AF0	2800
36	B40	2880
37	B90	2960
38	BE0	3040
39	C30	3120
40	C80	3200
41	CB0	3280
42	D20	3360
43	D70	3440
Status (Mod 4)	0	0

Bytes removed from the screen buffer are translated into characters using the following table:

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	nul	sp	0	&	à	ä	À	Ä	a	q	A	Q		P		
1	em	=	1	-	è	ë	È	Ë	b	r	B	R		S		
2	ff	'	2	.	ì	ï	Ì	Ï	c	s	C	S				Z
3	nl	"	3	,	ò	ö	Ò	Ö	d	t	D	T				
4	stp	/	4	:	ù	ü	Ù	Ü	e	u	E	U				
5	cr	\	5	+	ã	â	Ã	Â	f	v	F	V				
6		ı	6	¬	õ	ê	Õ	Ê	g	w	G	W				X
7		¦	7	−	ÿ	î	Y	Î	h	x	H	X				
8	>	?	8	°	à	ô	A	Ô	i	y	I	Y				
9	<	!	9		è	û	E	Û	j	z	J	Z				
A	[	$	ß	^	é	á	E	Á	k	æ	K	Æ				
B	]	¢	∮	∿	ì	é	I	É	l	ø	L	Ø				
C	)	£	#	¨	ò	í	O	Í	m	å	M	Å				
D	(	¥	@	`	ù	ó	U	Ó	n	ç	N	Ç				
E	}	Pts	%	´	ü	ú	Y	Ú	o	¯	O	;				
F	{	¤	_	~	ç	ñ	C	Ñ	p	¯	P	*				

For example, 2C is "#"; B8 is "Y"; B3 is "T". C0 to FF define indicator and attribute bytes. For more information, see "Attribute/EAB Characters."

Attribute/EAB Characters

Characters defining the type of data to be entered in a particular field are called attribute characters. When an attribute character is encountered, the character signals the controller and display that a particular field type is to follow. All data between that character and the next attribute character (indicating the beginning of the next field) is considered part of the field.

Attribute characters are normally displayed as blanks. If the attribute byte defines a field to be both protected and numeric, the cursor is positioned automatically by the controller to the next unprotected field (auto skip). You may display the attribute characters by entering <CTRL-F2>.

Attribute Characters define the following field characteristics:

- Start of a field.
- Whether a field is protected or unprotected. (A protected field cannot be modified by the operator; an unprotected field allows data entry.)
- Whether an input field (unprotected) accepts alphabetic or numeric, or both types of data.
- Whether the current field is to be displayed, not displayed, or intensified.
- Whether fields are detectable by light pen.
- Whether tab stops correspond to the first character of an unprotected field (auto skip).
- Modified Data Tag (MDT). (Using bit 0, the controller searches for modified fields. If the field has been modified, bit 0 is set to 1, and the controller updates that field accordingly.)

Extended Attribute Characters define the following field characteristics:

- Character type (normal, blink, reverse video, or underlined).
- Character color.
- Character set.

The DSI supports all programming definitions for attribute characters and the fields they define. Keyboard operations, such as CLEAR, or ENTER, function in the same manner as with 3278/79 terminals.

While the emulator program handles scan code translation, IRMA's 8X305 microprocessor handles 3270 controller protocol, such as actual data transfer, handshaking, and screen buffer maintenance. The 8X305 acts as a supervisor, assigning data entered to its proper position and function. It then sends this ordered data to the controller and, on return, sends the response to the proper source.

Proper interpretation of data from the buffer requires some understanding of attribute bytes. Normally, an attribute byte (byte from the last four columns, C0 to FF, of the character translation table) precedes, and a second follows, any field on the screen. The byte preceding the field defines how that field is handled. The data word bits for attribute characters in the screen buffer are defined as follows:

7 6 5 4 3 2 1 0	Bit Number
1 1 a b c c d e	Attribute Character

1,1 = Attribute identifier

a = 0 Unprotected
1 Protected

b = 0 Alphameric
1 Numeric

c,c = 00 Normal display, non-detectable
01 Normal display, detectable
10 Bright display, detectable
11 Non display, non-detectable

d = Reserved (Must always be zero)

e = Modified Data Tag

0 — Field has not been modified.
1 — Field has been modified by the operator.

The Extended Attribute Buffer (EAB) is subdivided into two types of attributes, Extended Field Attribute (EFA) and Extended Character Attribute (ECA). The EFA defines the field attributes while the ECA defines each character. The ECA is dependent upon the most recent EFA. When the ECA is 0, the attributes defined by the last EFA remain in effect. The following chart illustrates this relationship.

Extended Attribute Buffer	EFA GREEN	ECA 0	ECA 0	ECA BLUE	ECA 0	EFA RED	ECA 0
Attribute Buffer	UNPROT	A	B	C	D	PROT	E

Characters 'A' and 'B' are defined as 'green' and in an unprotected field. Character 'C' is still part of the same unprotected field but has been redefined as 'blue'. Character 'E' is in a protected field and is defined as 'red'. Characters 'A' and 'B' have an ECA value of 0, and their Extended Attribute characteristics revert to the most recent EFA. Character 'D' also has a value of 0. Its characteristics revert to the most recent EFA of unprotected and green. A non-zero ECA is a temporary redefinition of a character attribute and does not effect subsequent characters.

For the Extended Field Attributes (EFA) the data word bits are:

7	6	5	4	3	2	1	0	Bit Number
a	a	b	b	b	c	c	c	Extended Field Attributes

Bits a,a =
- 00 Normal Mode
- 01 Blink Character
- 10 Reverse Video Characters
- 11 Underline Character

Bits b,b,b =
- 000 Default to Base Color
- 001 Blue
- 010 Red
- 011 Magenta
- 100 Green
- 101 Cyan
- 110 Yellow
- 111 White

Bits c,c,c =
- 000 Base character set
- 001 APL
- 010 PS 2 (191 character)*
- 011 PS 3 (191 character)*
- 100 PS 4 (191 character)*
- 101 PS 5 (191 character)*
- 110 PS 6 (191 character)*
- 111 PS 7 (191 character)*

* not available with IRMA.

For the Extended Character Attributes (ECA), the data word bits are:

7 6 5 4 3 2 1 0			Bit Number
a a b b b c c c			Extended Field Attributes
Bits a,a	=	00	Reverts to most recent EFA
		01	Blink Character
		10	Reverse Video Characters
		11	Underline Character
Bits b,b,b	=	000	Reverts to EFA
		001	Blue
		010	Red
		011	Magenta
		100	Green
		101	Cyan
		110	Yellow
		111	White
Bits c,c,c	=	000	Reverts to EFA
		001	APL
		010	PS 2 (191 character)*
		011	PS 3 (191 character)*
		100	PS 4 (191 character)*
		101	PS 5 (191 character)*
		110	PS 6 (191 character)*
		111	PS 7 (191 character)*

When processing a screen buffer, the programmer must remember the most recent attribute byte encountered. Note that the screen size for attribute purposes is determined by the terminal model (1920 x 1 for Mod 2; 2560 x 1 for Mod 3; 3440 x 1 for Mod 4). Ends of lines play no part in attribute interpretation. Also, attributes are displayed on the screen as blanks.

BASICA subroutines are provided to handle line and field reads, keystroke sends, and status checks. These routines implement all necessary tabular character translations and proper handshaking. Using these subroutines is by far the easiest approach to this programming since all of the complicated functions are correctly handled. For a discussion of these subroutines, see Appendix C.

Appendix A

Error Conditions

Introduction

This section is designed to assist in correction of the most common problems encountered by IRMA users. The following paragraphs list the error message or screen condition associated with each problem, followed by a brief explanation of the problem that prompted the message or condition, and one or more suggestions for possible correction of the error.

2%%

Problem

Keyboard types do not match. The controller port is not configured for a TYPEWRITER style keyboard.

Solution(s)

1. To clear the error message from the screen, press the RESET (F10) key.
2. If this action does not clear the message, ensure that the controller port is configured for a 3278 type 2A typewriter style keyboard.

226 Error

Problem

A coaxial cable communication problem exists.

Solution(s)

1. Check for cuts, breaks, or any other damage to the coax cable.
2. Ensure that the coax cable is properly connected.

503 or 404 Error

Problem

Incorrect controller port configuration.

Solution(s)

Configure the controller port to match the device being used.

402 Error

Problem

The controller port is configured for other than the device being used (i.e. Mod 2 plugged into Mod 4, etc.).

Solution(s)

1. Enter RESET (F10) or CTRL/PAGE DOWN.
2. Reconfigure the controller port.

Insufficient Space for Screen Buffer

Problem

IRMA*link* file transfer was executed, and this message was received.

Solution(s)

1. The file must be an XEDIT file.
2. Verify that your on-board firmware is Revision 3.14 or above.

Program Too Large for Memory

Problem

The user may be running under DOS 1.0, not supported by IRMA. If the user receives this message while running under DOS 1.1 or 2.0, the program being executed requires more RAM storage than is available.

Solution(s)

1. Operate under DOS 1.1 or 2.0.
2. PC requires at least 128K RAM capacity.
3. The IRMA diskette is dual-density, double-sided. Your PC may be equipped with a single-sided disk drive. IRMA is available on single-sided diskettes; contact DCA.

Blank Screen with Blinking Cursor

Problem

Inactive controller port or faulty coax connection.

Solution(s)

Ensure that the controller port is active and that all connections are secure.

Invalid Response From Edit

Problem

IRMA*link* file transfer was executed, and this message was received from TSO EDIT or CMS XEDIT.

Solution(s)

1. Prior to execution of IRMA*link* file transfer, E78 must have been executed and TSO must have a Ready message, or CMS must be in the XEDIT mode.
2. If using TSO, ensure that the user ID profile matches the profile outlined in IRMA documentation.
3. If using CMS, an IRMA XEDIT data set must have been allocated with profiles as shown in the IRMA documentation.
4. Check the host file type. The file type must be valid as defined under TSO or CMS.

Appendix B

Using GENX

Introduction

GENX is a menu-driven program that allows you to create customized versions of the E78 emulator program, with specialized keyboard, color, and communications defaults. By using GENX, IRMA may be configured to accommodate features and function modes best suited to the system you are using.

GENX handles a variety of functions, including calibration of a light pen, enabling of COM1: inputs, setting of screen and keyboard types, and selection of PC "look-similar" modifications.

Only use GENX if you wish to customize E78 program. Otherwise, you can use IRMA without using the GENX option.

GENX Description

GENX allows you to select options that describe parameters of the system you are using. Selecting the appropriate options causes GENX to "patch" these options into E78. The next time IRMA is used, these selected options become the default parameters. If different or additional parameters are required for other users, a second version of E78 should be generated. Therefore, it is possible to have several customized versions of E78, each fulfilling the requirements of a specific application.

GENX is executable in two forms. One format is for the general user who wants to make use of the normal defaults without altering the keyboard mappings or altering E78 to accommodate unique requirements. The second format is for those users who require special configurations that are not included in the default selections.

The following discusses the first GENX format. If you wish to use the second GENX format, contact DCA Customer Support to obtain a copy of instructions on how to use the second format.

GENX Files

In order to run GENX, several files must be available in the current directory. These files are described below.

GENX.EXE	Actual GENX program which displays the menus and patches the program file
E78.MAP	Symbol table for the program to be modified. This file is produced by the DOS LINK program and contains entries which tell GENX how to find the various tables and switches used to apply E78 patches. This file is in a simple format and may be typed.
E78.GEN	Text file initially prepared by DCA containing information used by GENX to display menus and make user patches.

Executing GENX creates several files in the current directory. These files are listed below.

CE78.EXE	This is your customized version of E78. This name is used to execute a customized E78 program.
CE78.LOG	Text file in a format similar to that of the E78.GEN file described above. The CE78.LOG file contains a listing of all patches installed in the CE78.EXE file.

For general use, the E78GEN.BAT file contains instructions for starting GENX. This produces the customized E78 program, CE78.EXE.

These same files are also used by the second GENX format. For the second format, you must perform some file modification.

For example, to create a special keyboard modification not listed among the default settings, the file E78.GEN must be modified by the user prior to generation of the actual CE78.EXE program. However, for most applications, the first format is adequate for creating the customized version of E78.

Using GENX

If not already accomplished, you should make duplicate copies of the E78 diskette and label them as working copies. Store the original E78 diskette in a safe place for later use.

WARNING: Do not modify the original diskette. Make all modifications to the working copies of the E78 diskette.

All modifications must be made while operating in PC-DOS. The following steps explain how to begin using GENX.

STEP 1: **Insert PC-DOS diskette into Drive A**

STEP 2: **Turn on your computer**

STEP 3: **Enter the current date**

STEP 4: **Enter the current time**

The DOS prompt A> should appear. Replace the PC-DOS diskette in drive A with your IRMA diskette.

STEP 5: **Enter E78GEN** To begin running GENX

The GENX Main Menu

Following display of copyright, trademark, and revision information, the following main menu is displayed.

```
E78 Terminal Emulator Customization Menu

  1 — Disable 26th line status display
  3 — Make emulator auto-resident
  5 — Make 2 color mode default
  6 — Make 7 color mode default
  8 — Make SHOW COLUMNS default
 10 — Convert to pre-1.20 keyboard
 12 = Select KEYBOARD & SCREEN type
 13 = Set LIGHT PEN corrections
 14 = PC look-alike patches
 15 = Setup COM1: input parameters
 17 — Disable NON-DISPLAY display
 18 — Force dark blue to cyan
 20 — Select IRMAlette operation

                                  99 = Exit GENX program
Your selection:
```

Menu Options

The following table describes menu options 1 through 10. These options are complete as listed, without submenus.

Menu items with submenus are described on subsequent pages with menu screens shown.

Option	Description
1	Disables the display of the status line (line 26)
3	Causes the customized E78 program to become resident upon power-up
5	Causes the 4-color mode to become the default
6	Causes the 7-color mode to become the default
8	Causes the unprotected null fields to be filled with dots, showing the field size
10	Causes conversion to the pre-1.20 keyboard. Should not be selected if you expect to later use the non-typewriter keyboards (e.g., APL, TEXT, or DATA ENTRY).

The following describes GENX menu options 12 thru 15.

KEYBOARD AND SCREEN Type — Option 12

Menu option 12 allows the selection of KEYBOARD and SCREEN types. Changes to keyboard and screen types may require changes to coax configuration. The following submenu appears if you select option 12:

```
E78 Terminal Emulator Customization Menu

                Select KEYBOARD & SCREEN type

1  — Typewriter [default]          17 — No attached keyboard
2  — Typewriter w/ numeric lock    18 — Reserved 0000
3  — Typewriter, PSHICO            19 — Reserved 0011
                                   20 — Reserved 1011
5  — APL
6  — APL w/ numeric lock           22 — Mod 2 Screen [24x80]
7  — APL, PSHICO                   23 — Mod 3 Screen [32x80]
                                   24 — Mod 4 Screen [43x80] [8k]
9  — Text
10 — Text w/ numeric lock
12 — Data Entry I
13 — Data Entry I w/ numeric lock
14 — Data Entry
15 — Data Entry II w/ numeric lock
98 = Return to previous menu       99 = Exit GENX program
Your selection:
```

Item 12 presents options for selecting all possible keyboard configurations. You should be aware, however, that selection of a specific keyboard does not necessarily mean that the resulting keyboard configuration is useful. For example, TEXT and APL keyboards should not be selected unless an extended character set is installed in the display adapter board. If either the TEXT or APL keyboard is selected without installation of the character set, special characters are displayed as musical notes and game symbols.

In addition, selection of either DATA ENTRY keyboards moves the numeric and PF keys into unusual locations. To make DATA ENTRY keyboards workable, you must make several decisions. These decisions are based upon the specific applications involved, as well as upon the layout of the PC keyboard versus that of the 3278 keyboard. If DATA ENTRY keyboards are essential to the application, the DCA Technical Support Group will assist you in the construction of a GENX menu item appropriate to a specific application.

LIGHT PEN Correction — Option 13

Menu option 13 is used for light pen correction. The option should be used only when pressing the light pen against the edge of a screen field causes the adjacent field to be selected. If this occurs on the left edge of the field, use the MINUS (–) selections. If it occurs on the right edge of the field, use the PLUS (+) selections to calibrate the light pen. If none of the available selections result in correct light pen operation, the pen may be defective or internally maladjusted. To enable the light pen, attach the pen to the display adapter board.

The following submenu appears if you select option 13:

```
E78 Terminal Emulator Customization Menu
                        Set LIGHT PEN correction

1 — -3  X..o
2 — -2  .X.o
3 — -1  ..Xo
4 — >0     o
5 — +1     oX..
6 — +2     o.X.
7 — +3     o..X

98 = Return to previous menu          99 = Exit GENX program
```

PC Look-Alike Patches — Option 14

Menu option 14 is used for PC look-alike patches. The following submenu appears if you select option 14:

```
E78 Terminal Emulator Customization Menu

                    PC look-alike patches

1 — Eagle 1600 screen cleanup
3 — Force Horizontal update sync
5 — Disallow 26 line color display
7 — Always set cursor shape on exit
98 = Return to previous menu                99 = Exit GENX program

Your selection:
```

The following table describes available submenu options for option 14:

Option	Description
1	Eliminates the dots that sometimes appear on the screen. Also enables "8086"-type screen access. Should be selected when IRMA/E78 is used in any 8086-based machine.
3	Necessary to remove or reduce screen flicker on some non-IBM machines, notably the EAGLE PC
5	Selected when using the COMPAQ color/monochrome display
7	If you are using a monochrome card, this option will set your cursor shape to an underline character every time you exit E78. If you do not select this option, the cursor shape may be a dash.

Setup COM1 — Option 15

Menu option 15 is used to set up COM1: input parameter. The following submenu appears if you select option 15:

```
E78 Terminal Emulator Customization Menu
Setup COM1: input parameters
1 — Enable COM1: character input
2 — Disable COM1: key clicks
4 — Select 4800 baud
5 — Select 2400 baud
6 — Select 1200 baud
7 — Select  300 baud
8 — Select  110 baud
98 = Return to previous menu                    99 = Exit GENX program
Your selection:
```

The following table describes available submenu options for option 15.

Option	Description
1	Enables asynchronous character input (such as barcode readers)
2	Disables key clicks

Options 4-8 establish the baud rate for the device attached to COM1.

4	4800 baud
5	2400 baud
6	1200 baud
7	300 baud
8	110 baud

Main Menu Options 17, 18, and 20

The following describes main menu options 17, 18, and 20:

Option	Description
17	When CTRL & F2 are pressed, IRMA will display its current attributes on your PC screen. A "non-display" field will show attributes which are not displayed during normal operation, such as your password. For security reasons, displaying this information may not be acceptable. This option will disable the non-display field while allowing you to view other IRMA attributes.
18	Makes the dark blue on your screen lighter and brighter, thus making it easier to read.
20	This option must be selected if you are using IRMA*lette*. For more information, see the IRMA*lette* User's Manual.

Auto Residency

Without the auto-resident feature, you must type CE78 <RETURN> each time you need to access the 3270 mode (unless you make CE78 resident by pressing <CTRL-HOME>). However, if you select option 3 from the GENX main menu, "Make emulator auto-resident", you can automatically alternate between 3270 mode and PC mode by simply pressing both SHIFT keys at the same time.

To install auto-residency, select option 3 from the GENX main menu. To execute the auto-resident version, type CE78 <RETURN> from the DOS prompt. Copyright, trademark, revision information, and USER CUSTOMIZED VERSION is displayed. At this point, you can enter 3270 mode by pressing both SHIFT keys simultaneously, (pressing both SHIFT keys again returns you to PC mode).

You must type CE78 <RETURN> each time the PC is "booted." However, the CE78 command may be placed in the AUTOEXEC. BAT file. This causes the CE78 command to execute automatically whenever the PC is booted.

Appendix C

BASICA Subroutines

Introduction

The IRMA screen buffer is accessed by entering keystrokes from your terminal, as in in normal use. The IRMA screen buffer may be accessed by user-written programs as well. Since individual user needs are often quite varied, no single comprehensive PC program is appropriate for all users. However, predictable data handling routines have been formulated into subroutines for use in BASICA programs.

The BASICA subroutines discussed in this section provide the foundation for automatic data transfer via IRMA. Programmers may access the appropriate routines for many of the complicated functions associated with passing data to and from the host. The subroutines allow the generation of keystrokes, as well as the reading and writing of screens from within a user program. The routines may also serve as a model for the programmer who wishes to access IRMA in PASCAL, FORTRAN, and other languages.

Prior to using any of the routines in this section, programmers should refer to Chapter 5, "Technical Reference" as well as the IBM PC Technical Reference, IBM GA27-2849 3270 Information Display System Configuration, and documentation related to the mainframe application with which the user wishes to communicate.

To speed execution of an individual program, DCA recommends use of the Compiler version of Microsoft® BASIC. However, for ease of the debugging process, we recommend initial programming in the interpretive BASIC forms of Advanced or Disk BASIC.

Features

- May be used with BASIC, BASICA, and the IBM BASIC complier.
- Access to screen data by row/column or field number.
- Ability to read, modify, or write unprotected fields.
- Automatic code conversions between ASCII and 327X display buffer.
- Full support of total ASCII character set as defined in the IBM PC Technical Reference.
- Ability to check data type when modifying screen fields.
- Easy modification of character translation tables for support of unique user applications.
- Fully commented sources [PC78SUBS.LST] for detailed examination or maintenance.
- Additional sparsely commented source [PC78SUBS.BAS] for reduced storage requirements in large-user applications.
- Ability to access any screen information, including Attributes and Extended Attribute Buffer (EAB).
- Easy access to IRMA status information, including alarm status, cursor position, and line sync indicator.
- Capability of triggering user application program action on screen updates from the host computer.

Before using any of the subroutines described on the following pages, an initialization call must be made. This call causes all parameters to be set to their starting values. The initialization call must be made prior to the use of any other routine. Include at the beginning of your program:

```
GOSUB 50000
```

This appendix shows each routine in two ways: name and statement number. BASICA is able to call routines by statement number only; however, some users find the use of routine names convenient. Note that each routine begins with a REMark statement that includes the proper name of the routine.

Routine descriptions include variables to which the user must assign values prior to calling the routine (Input), as well as variables assigned values by the routine itself (Output). All input variables MUST be given values by the user. Reserved variables are listed at the end of this document.

In addition, this appendix contains a sample BASICA program illustrating the use of each routine. This program may be used as a guideline for user-written data-handling programs.

NOTE: The sample programs provided are samples only. No attempt should be made to execute the programs.

KEYS

Keystroke Send

Statement Number:	50600	
Routine Name:	KEYS	
Input Variables:	I.VST$	String to be sent to host
Output Variables:	I.VER%	Error status (device or key timeout)

Remarks: This routine sends strings as keystrokes to the host computer. The string length is confined to BASICA's limit of 255 characters.

NOTE: See Reserved Names section for specification of error status values.

Reference: In Appendix A, the sample program, SAMPLES.BAS, lines 600 - 620, provides an example of sending keystrokes. Line 602 sets the value for the input variable; line 604 calls the Keystroke Send subroutine. This subroutine occupies lines 50600 - 50660 of the same sample program. Line 612 sets the input variable to a new value and line 614 calls the Keystroke Send subroutine.

How To Use: To send a keystroke to the mainframe computer, the program input variable I.VST$ must be set equal to the string representing the keystrokes to be sent. For example, to send the characters ABCD, the program would read:

```
I.VST$ = "ABCD"
GOSUB 50600
```

When the keystroke characters to be sent are other than simple characters (A, B, C, ; etc.), as in Function 1 (F1) or Control C (CTRL C), these special characters are represented by the string function CHR$(n), where CHR$ indicates a special character to follow, and n is the ASCII code value of the character. For example, to send the characters ABCD and a <RETURN>, the program reads:

```
I.VST$ = "ABCD" + CHR$(13)
GOSUB 50600
```

In this example, 13 is the ASCII code value of carriage return.

For certain keys or key combinations that cannot be represented in standard ASCII code (PF and PA keys, for example), the characters to be sent are represented by Extended ASCII code values. In such cases, the input variable I.VST$ must be set equal to the string function CHR$(00) = CHR$(n). In this string, 00 indicates that extended characters follow, and n represents the Extended ASCII code value of the key combination needed to invoke the desired function.

For example, consider the PF1 key. The PF1 key is invoked by striking ALT 1 (see keyboard layout). The Extended ASCII code value of ALT 1 is 120 (ASCII code values and extended ASCII code values can be found in appendix G of the IBM BASIC manual.)

Thus, the program would read:

```
I.VST$ = CHR$(00) + CHR$(120)
GOSUB 50600
```

In another example, the PA2 key is invoked by striking CTRL K. The Extended ASCII code value of CTRL K is 37. Thus, to send PA2, the program reads:

```
I.VST$ = CHR$(00) + CHR$(37)
GOSUB 50600
```

Please note that this routine would not be used by itself. It is used in conjuction with several others. This explanation was provided to clarify the use of standard ASCII characters and the extended ASCII characters used by the IBM PC. Please examine the sample program provided to see how this routine uses and is used by other routines.

FIND

Find Unprotected Field

Statement Number:	50700	
Routine Name:	FIND	
Input Variables:	I.VFL%	Field number to be found
Output Variables:	I.VER%	Error status (device timeout, field error)
	I.VCB%	Pointer to leading attribute
	I.VCE%	Pointer to trailing attribute
	I.VFS%	Current field length
	I.VRO%	Row address of field data
	I.VCL%	Column address of field data
	I.BUF% (0)	Leading attribute character

Remarks: This routine searches IRMA's internal screen buffer for unprotected fields. As it searches, each unprotected field is counted until the specified count is reached. The absolute address of the selected field's unprotected attribute is returned in I.VCB%. The field contents are then scanned, counting characters and searching for another attribute. I.VCE% is left as an absolute pointer to the trailing attribute. I.VCB% and I.VCE% are used internally for the "Find Next" routine and for several field type consistency checks. The following variables SHOULD NOT be modified by the user: I.V CB%, I.VFS%, I.VCE%, I.BUF%. They must retain their values which were assigned by the FIND and FNEXT subroutines, since they provide the link between these subroutines as well as between successive FNEXT subroutine calls.

Reference: In the sample program, SAMPLES.BAS, provided in the Appendix, note lines 200 – 240. This portion of the sample program finds and prints each field. Line 202 sets the value for I.VFL% to one to find the first field. Line 210 calls the subroutine, FIND. The next portion of the program reads the found field and prints the specified information. Line 240 calls the subroutine which finds the next field.

FNEXT

Find Next Field

Statement Number:	50800	
Routine Name:	FNEXT	
Input Variables:	I.VFL%	Field number to be found
Output Variables:	I.VER%	Error status (device timeout, field error)
	I.VCB%	Pointer to leading attribute
	I.VCE%	Pointer to trailing attribute
	I.VFS%	Current field length
	I.VRO%	Row address of field data
	I.VCL%	Column address of field data
	I.BUF% (0)	Leading attribute character

Remarks: This routine increments I.VFL%, initializes the internal variables, and goes to the FIND routine.

Reference: For an example of usage, see lines 200 – 240 of SAMPLES.BAS listed in the Appendix. Line 230 calls FNEXT. This routine can be used only after FIND has been used, as it depends on the value of I.VFL% specified in FIND.

RDFLD

Read Field Contents

Statement Number:	50900	
Routine Name:	RDFLD	
Input Variables:	None	
Output Variables:	I.VER%	Error status (device timeout)
	I.BUF% ()	PC buffer is filled with field contents. I.BUF%(0) contains the leading attribute character. I.BUF%(1) through I.BUF% (I.VFS%) contain screen data and EAB information. Screen data is contained in the low-order byte, and EAB information is in the high-order byte.

Remarks: RDFLD transfers IRMA's internal screen memory of screen data and EAB contents to an internal array within the PC. FIND and FNEXT must be called to setup buffer pointers before calling RDFLD.

Reference: RDFLD is also used in the lines 200 – 240 of SAMPLES.BAS.

WRFLD

Write Field

Statement Number:	51000	
Routine Name:	WRFLD	
Input Variables:	I.BUF% ()	Internal screen buffer
	I.VCB%	Initial attribute pointer
	I.VFS%	Field length
Output Variables:	I.VER%	Error status (device timeout, illegal character)

Remarks: This routine writes the contents from the PC's internal buffer to IRMA's screen memory. WRFLD transfers data to the screen memory with error checking appropriate to the 3270 terminal system. The field is verified to be non-protected, and only I.VFS% characters are written. If a non-numeric character is found in a numeric only field, the routine aborts without modifying the screen memory. This routine must be preceded by a FIND.

Reference: SAMPLES.BAS, lines 300 – 345, gives an example of how one might use this subroutine. This portion of the program finds a field, and puts a string within that field into the PC's buffer. WRFLD writes the contents of the PC's buffer into IRMA's internal screen memory.

GTSTR

Get String

Statement Number:	51100	
Routine Name:	GTSTR	
Input Variables:	I.BUF% ()	Internal screen buffer
	I.VFS%	Field length
	I.VOO%	Offset within field to begin transfer
Output Variables:	I.VER%	Error status (Offset out of bounds)
	I.VST$	ASCII data recovered from buffer
	I.VOO%	Offset to REMAINDER of field (If field is longer than 254 characters)

Remarks: The GTSTR routine retrieves the field data from the PC's internal buffer and converts the characters to the extended ASCII used by the PC in its display buffer.

Reference: An example of this subroutine is found in lines 200-240 of the sample program. GTSTR converts IRMA's buffer code to the extended ASCII character set used by the PC's display buffer. To transfer data from one buffer to another, this conversion must take place before the string can be read.

PUSTR

Put String

Statement Number:	51200	
Routine Name:	PUSTR	
Input Variables:	I.VST$	String to be placed in buffer
	I.VOO%	Offset in buffer at which to begin.
Output Variables:	I.VER%	Error status (device timeout)
	I.VOO%	Offset to remainder of field
	I.BUF%	Screen format buffer

Remarks: PUSTR writes an ASCII string into the PC's internal display buffer. The routine moves the ASCII string into the PC's internal buffer and converts the string to 3270-type buffer codes. This routine should be called prior to a WRFLD in order to place the data to be written in the internal buffer.

Reference: An example of this subroutine is found in lines 300-345 of SAMPLES.BAS. PUSTR is used here in combination with FIND and FNEXT. FIND locates the field and PUSTR translates the ASCII string, I.VST$, into IRMA's buffer code and that string to the internal buffer (I.BUF%). WRFLD takes I.BUF% and writes it in the screen buffer.

RDABS

Read Absolute Screen

Statement Number:	51300	
Routine Name:	RDABS	
Input Variables:	I.VRO%	Row number of starting character
	I.VCL%	Column number of starting character
	I.VRR%	Length of area to read
Output Variables:	I.VER%	Error status (device timeout)
	I.VRO%	Row position after last character read
	I.VCL%	Column position after last character read
	I.VST$	ASCII form of screen data
	I.VS0$	EAB data

Remarks: This routine moves IRMA's screen memory characters and EAB data into user strings from a given row and column screen position, for a given length. Screen characters are translated into ASCII. EAB data is unmodified.

Reference: The example for this subroutine is found in lines 400-412 of SAMPLES.BAS. Variables are set for the row and column at which the read is to begin, and the number of

columns (characters) to be read before calling the subroutine in line 406. In this particular example, the procedure is performed five times with the first 40 characters (I.VRR%) of the top five lines (I.VRO%) of the screen displayed. (Value for I.VRO% is determined by the FOR/NEXT loop, lines 404 and 412.)

GTCP

Get Cursor Position

Statement Number:	51400	
Routine Name:	GTCP	
Input Variables:	None	
Output Variables:	I.VER%	Error status (device timeout)
	I.VRO%	Cursor row position
	I.VCL%	Cursor column position

Remarks: GTCP reads the 3270 screen cursor position from IRMA. It should be noted that the row address may not be within the confines of the normally displayed screen.

Reference: Lines 500-520 of SAMPLES.BAS provides an example of GTCP. This portion of the program retrieves the cursor position and prints the location of the buffer pointers. It also calls a subroutine listed in lines 900-980, which retrieves and displays the state of the main and aux status words.

XPOR

Execute Power-on-reset

Statement Number:	50100	
Routine Name:	XPOR	
Input Variables:	None	
Output Variables:	I.VER%	Error status (device timeout)

Remarks: The XPOR routine causes the controller to set the terminal as though it has just been powered up. This call can be useful in clearing some controller errors, especially if the coaxial cable has been disconnected.

Reference: This subroutine is not used in the sample program; however, the subroutine is listed in lines 50100-50108. Its use should be limited to particular conditions, such as during data transfer if the controller has sent data which does not appear to be that which was requested. When all other attempts to remedy the problem have been exhausted, the insertion of this routine into the program resets the terminal. The controller acknowledges the terminal as if the terminal has just been powered on.

GSTAT

Get Status

Statement Number:	50200	
Routine Name:	GSTAT	
Input Variables:	None	
Output Variables:	I.VST%	Main device status
	I.VAX%	Auxiliary device status
	I.VER%	Error code

Remarks: GSTAT reads the current main and aux status from IRMA. Each bit in these status words has a specific meaning. Parameters are provided (See the list at the end of this section) to a user to 'and' mask the status with a parameter to test for a specific condition. For example, to test for the buffer modified status (I.MDI% from table), the program would read:

```
GOSUB  50200
IF I.VST% AND I.MDI% THEN GOTO
```

Bit mask parameters for both I.VST% and I.VAX% are provided.

Reference: Lines 700-714 contain an example of this subroutine. The routine reads the status, displays the current status, clears each of the status bits, and then reads and displays the status after the clear.

RSTAT

Reset Status

Statement Number:	50300	
Routine Name:	RSTAT	
Input Variables:	I.VST%1	Bits in status word to be reset
Output Variables:	I.VER%	Error status (device timeout)

Remarks: RSTAT resets status bits in the main status word. If a status bit such as I.MPR% (controller reset) is read in a GSTAT, the user program must take any action needed, then reset the bit. This can be done by calling RSTAT with I.VST% as set by GSTAT. Note that any bit set in I.VST% when RSTAT is called will be reset.

Reference: Lines 700-714 include an example of this subroutine. After displaying the status, the status bits are cleared or reset.

STDNM

Set Trigger Data AND Mask

Statement Number:	50400	
Routine Name:	STDNM	
Input Variables:	I.VRO%	Row for trigger test
	I.VCL%	Column for trigger test
	I.VMS%	Trigger mask value
	I.VVL%	Trigger test value
Output Variables:	I.VER%	Error code on exit

Remarks: STDNM allows the program to declare a location on the screen, for a delay, until a specific value is written into that location. WTRIG does not return until that value is found in the requested location. The location = value test is made only on those bits which have a mask value of 1. Thus, to make an exact match test, the value of I.VMS% must be set to 255 (decimal), which is all 1 bits. The condition of I.VMS% = 0 is special. If the mask is 0, any CHANGE in the location requested results in the return of WTRIG. In this case, I.VVL% is unused.

Reference: Lines 800-899 contain an example of this routine. Lines 802 and 804 set the input variables. The row and column are set to (1,1) which positions the pointer to the upper left corner. Mask and text values are both set to 0, which results in a trigger whenever any change occurs in 1, column 1.

WTRIG

Wait Trigger

Statement Number:	50500	
Routine Name:	WTRIG	
Input Variables:	I.VTO%	Time constant (in seconds)
Output Variables:	I.VER%	Error code
	I.VTO%	Time remaining

Remarks: WTRIG waits until a specific trigger event occurs (see STDNM). The time constant, I.VTO, allows the programmer to select how long (in seconds) WTRIG waits for the event to become true.

Reference: This subroutine is used in conjunction with STDNM. The WTRIG routine is called in line 824. It allows a program to wait for a trigger condition specified by STDNM to occur. If I.VTO% returns with a value of 0, no trigger event has occurred, and the call to WTRIG has timed out.

Reserved Names

The BASICA subroutines occupy statement numbers from 50000 to 59999. No user statements should be included in this range. The variables, flags, and parameters used by these routines are named to prevent conflict by user routines. The following list illustrates this usage.

IRMA commands

Name	Use	Type
I.CRD%	Slave read data	Parameter
I.CWR%	Slave write data	Parameter
I.CAC%	Slave aux status and cursor update	Parameter
I.CCL%	Slave main status bit clear	Parameter
I.CKY%	Slave send keystroke	Parameter
I.CSP%	Slave selector pen strike	Parameter
I.CXP%	Slave execute power-on-reset	Parameter
I.CMD%	Slave load trigger mask and data	Parameter
I.CTA%	Slave load trigger address	Parameter
I.CIM%	Slave set attention request mask	Parameter

Main status word (I.VST%) bit mask

Name	Use	Type
I.MAX%	Auxiliary status change	Mask
I.MTG%	Trigger event occurred	Mask
I.MKY%	Slave key scan code buffer empty	Mask
I.MXX%*	Unused, reserved for future use	Mask
I.MPR%	Controller requested reset occurred	Mask
I.MCC%	Last command complete flag	Mask
I.MDI%	IRMA buffer modified by controller	Mask
I.MCM%	Cursor position modified by controller	Mask

Auxiliary status word (I.VAX%) bit mask

Name	Use	Type
I.MXX%*	Unused, reserved for future use	Mask
I.MPO%	IRMA polled since last status read	Mask
I.MAL%	Sound alarm request	Mask
I.MDD%	Display disabled (inhibit) request	Mask
I.MCI%	Cursor inhibited	Mask
I.MRC%	Reverse block cursor select	Mask
I.MBC%	Blinking cursor select	Mask
I.MCK%	Keyboard clicker enabled	Mask

*Known to be duplicates—Reserved

IRMA device code parameters

Name	Use	Type
I.RG0%	Device code of register 0	Parameters
I.RG1%	Device code of register 1	Parameters
I.RG2%	Device code of register 2	Parameters
I.RG3%	Device code of register 3	Parameters
I.RST%	Handshake slave to start select	Parameters
I.RAK%	Handshake ATTN acknowledge	Parameters
I.RAF%	Handshake flag read select	Parameters

IRMA handshake flag bit masks

Name	Use	Type
I.MAT%	IRMA requests System Unit attention	Mask
I.MBS%	IRMA busy with System Unit request	Mask

BASICA internal use temporary variables

I.VT0%	I.VT1%	I.VT2%	I.VT3%	I.VT4%	Temp. variable
I.VT5%			I.VT8%	I.VT9%	Temp. variable
I.VS0$	I.VS1$	I.VS2$	I.VT3$	I.VS4$	Temp. variable
I.VT5$	I.VS6$	I.VS7$	I.VS8$	I.VS9$	Temp. variable

BASICA subroutine I/O variables

Name	Use	Type
I.VER%	Error code returned to user 0—No error occurred 1—IRMA did not respond to command 2—Row number out of range 3—Column number out of range 4—Byte value out of range 5—Invalid field type for operation 6—Invalid character in NUMERIC only field 7—Field number out of range 8—Invalid extended key code 9—Timeout on key send attempt 10—Timeout on trigger wait event 11—Illegal internal buffer pointer 12—Field or string too long 13—Found field does not match internal buffer type 14—Buffer offset out of range of internal buffer 15—Bad key scan code	Variable
I.VST%	Main status word	Variable
I.VRO%	Screen row number (0-24 [0 is the status line, row 1 starts at the top of the screen])	Variable
I.VCL%	Screen column number (1-80)	Variable
I.VMS%	Trigger mask byte	Variable
I.VVL%	Trigger value byte	Variable
I.VFG%	General Boolean flag	Variable
I.VST$	General string variable	Variable
I.VFL%	Current field length	Variable
I.VAX%	Aux status value	Variable
I.VOO%	Buffer offset pointer for string I/O	Variable
I.VRR%	Raw screen read data length	Variable
I.VCB%	Internal pointer to beginning of field	Variable
I.VCE%	Internal pointer to end of field	Variable
I.VFS%	Internal field length (size)	Variable
I.VTO%	Timeout constant	Variable
I.TAB%	Code conversion tables	Variable
I.BUF%	Screen format buffer	Variable

The aforementioned parameters are constants. BASICA does not provide for any parametric declarations; therefore, all basic subroutines use one set of variables, which are initialized to the correct value. Parameters NEVER change during a program execution.

The temporary variables listed above are used by these subroutines to hold values needed during execution. No data is guaranteed to be left in any of these variables.

The argument-passing variables are used to pass data to and/or from the BASICA subroutines. In the description of each routine, those variables listed as INPUT must be set prior to the GOSUB. Variables listed as output are updated during routine execution. Note that some variables are both input and output.

The user is warned that these routines provide the protection needed to prevent the sending of illegal data to the 3270 network. This protection is only available if the routines remain unmodified.

```
100 PRINT "Building IRMATABS.OVR for IRMASUBS package"
110 DEFINT A-Z
115 DEF SEG
120 DIM I.TAB%(1279)                                    256+256+512+256
130 RESTORE
135 FOR I.VT0%=0 TO 1279
140   READ I.TAB%(I.VT0%)
145 NEXT
200 BSAVE "IRMATABS.OVR",VARPTR(I.TAB%(0)),2560 ' Save the whole array
210 PRINT "Build complete. "
220 NEW
300 REM Offsets into I.TAB% are as follows:
302 REM
304 REM      000 - ASCII to buffer code table     (256 entrys)
306 REM      256 - Buffer code to ASCII table     (256 entrys)
308 REM      512 - Normal keycodes to key no.     (256 entrys)
310 REM      768 - Extended keycodes to key no.   (256 entrys)
312 REM     1024 - Key number to scan code        (256 entrys)
320 REM
59600 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000 : '*ASCII
59601 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59602 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59603 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59604 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59605 DATA &h0000, &h0000, &h0010, &h0019, &h0013, &h002C
59606 DATA &h001A, &h002E, &h0030, &h0012, &h000D, &h000C
59607 DATA &h00BF, &h0035, &h0033, &h0031, &h0032, &h0014
59608 DATA &h0020, &h0021, &h0022, &h0023, &h0024, &h0025
59609 DATA &h0026, &h0027, &h0028, &h0029, &h0034, &h00BE
59610 DATA &h0009, &h0011, &h0008, &h0018, &h002D, &h00A0
59611 DATA &h00A1, &h00A2, &h00A3, &h00A4, &h00A5, &h00A6
59612 DATA &h00A7, &h00A8, &h00A9, &h00AA, &h00AB, &h00AC
59613 DATA &h00AD, &h00AE, &h00AF, &h00B0, &h00B1, &h00B2
59614 DATA &h00B3, &h00B4, &h00B5, &h00B6, &h00B7, &h00B8
59615 DATA &h00B9, &h001B, &h0015, &h0016, &h0036, &h002F
59616 DATA &h003D, &h0080, &h0081, &h0082, &h0083, &h0084
59617 DATA &h0085, &h0086, &h0087, &h0088, &h0089, &h008A
59618 DATA &h008B, &h008C, &h008D, &h008E, &h008F, &h0090
59619 DATA &h0091, &h0092, &h0093, &h0094, &h0095, &h0096
59620 DATA &h0097, &h0098, &h0099, &h000F, &h0017, &h000E
59621 DATA &h003B, &h0000, &h0000, &h0000, &h0000, &h0000
59622 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59623 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59624 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59625 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59626 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59627 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59628 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59629 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59630 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59631 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59632 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59633 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59634 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59635 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59636 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59637 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59638 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59639 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59640 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59641 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59642 DATA &h0000, &h0000, &h0000, &h0000
59700 DATA &h0020, &h0020, &h0020, &h0020, &h0020, &h0020 : '*EBCDIC
59701 DATA &h0020, &h0020, &h003E, &h003C, &h005B, &h005D
59702 DATA &h0029, &h0028, &h007D, &h007B, &h0020, &h003D
59703 DATA &h0027, &h0022, &h002F, &h005C, &h005D, &h007C
59704 DATA &h003F, &h0021, &h0024, &h005B, &h006C, &h0079
59705 DATA &h0070, &h006F, &h0030, &h0031, &h0032, &h0033
59706 DATA &h0034, &h0035, &h0036, &h0037, &h0038, &h0039
59707 DATA &h0062, &h0073, &h0023, &h0040, &h0025, &h005F
```

```
59708 DATA &h0026, &h002D, &h002E, &h002C, &h003A, &h002B
59709 DATA &h005E, &h005F, &h002E, &h0020, &h005E, &h007E
59710 DATA &h0022, &h0060, &h0027, &h0035, &h0061, &h0065
59711 DATA &h0069, &h006F, &h0075, &h0061, &h006F, &h0079
59712 DATA &h0061, &h0065, &h0065, &h0069, &h006F, &h0075
59713 DATA &h0075, &h0063, &h0061, &h0065, &h0069, &h006F
59714 DATA &h0075, &h0061, &h006F, &h0079, &h0061, &h0065
59715 DATA &h0065, &h0069, &h006F, &h0075, &h0075, &h0063
59716 DATA &h0041, &h0045, &h0049, &h004F, &h0055, &h0041
59717 DATA &h004F, &h0059, &h0041, &h0045, &h0045, &h0049
59718 DATA &h004F, &h0055, &h0059, &h0043, &h0041, &h0045
59719 DATA &h0049, &h004F, &h0055, &h0041, &h0045, &h0049
59720 DATA &h004F, &h0055, &h0041, &h0045, &h0049, &h004F
59721 DATA &h0055, &h004E, &h0061, &h0062, &h0063, &h0064
59722 DATA &h0065, &h0066, &h0067, &h0068, &h0069, &h006A
59723 DATA &h006B, &h006C, &h006D, &h006E, &h006F, &h0070
59724 DATA &h0071, &h0072, &h0073, &h0074, &h0075, &h0076
59725 DATA &h0077, &h0078, &h0079, &h007A, &h0061, &h006F
59726 DATA &h0061, &h0063, &h003B, &h002A, &h0041, &h0042
59727 DATA &h0043, &h0044, &h0045, &h0046, &h0047, &h0048
59728 DATA &h0049, &h004A, &h004B, &h004C, &h004D, &h004E
59729 DATA &h004F, &h0050, &h0051, &h0052, &h0053, &h0054
59730 DATA &h0055, &h0056, &h0057, &h0058, &h0059, &h005A
59731 DATA &h0041, &h004F, &h0041, &h0043, &h003B, &h002A
59732 DATA &h0020, &h0020, &h0020, &h0020, &h0020, &h0020
59733 DATA &h0020, &h0020, &h0020, &h0020, &h0020, &h0020
59734 DATA &h0020, &h0020, &h0020, &h0020, &h0050, &h0053
59735 DATA &h0041, &h001E, &h0042, &h0036, &h0010, &h0016
59736 DATA &h001A, &h00E9, &h0006, &h0001, &h0042, &h0003
59737 DATA &h00A8, &h00DB, &h0020, &h0020, &h0020, &h0020
59738 DATA &h0020, &h0020, &h0020, &h0020, &h0020, &h0020
59739 DATA &h0020, &h0020, &h0020, &h0020, &h0020, &h0020
59740 DATA &h0015, &h0017, &h005A, &h005F, &h0009, &h000A
59741 DATA &h0058, &h0016, &h001B, &h0025, &h00FB, &h00B7
59742 DATA &h0034, &h0041, &h00E9, &h0002
59800 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000 : '*KEYCODE
59801 DATA &h0000, &h0000, &h1000, &h1500, &h3400, &h0000
59802 DATA &h0000, &h4C00, &h0000, &h0000, &h0000, &h0000
59803 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59804 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59805 DATA &h0000, &h0000, &h4A00, &h2039, &h3239, &h0639
59806 DATA &h0739, &h0839, &h0A39, &h3200, &h0C39, &h0D39
59807 DATA &h0B39, &h0F39, &h4200, &h0E00, &h4300, &h4400
59808 DATA &h0D00, &h0400, &h0500, &h0600, &h0700, &h0800
59809 DATA &h0900, &h0A00, &h0B00, &h0C00, &h3139, &h3100
59810 DATA &h3A00, &h0F00, &h3A39, &h4439, &h0539, &h2839
59811 DATA &h3F39, &h3D39, &h2A39, &h1839, &h2B39, &h2C39
59812 DATA &h2D39, &h1D39, &h2E39, &h2F39, &h3039, &h4139
59813 DATA &h4039, &h1E39, &h1F39, &h1639, &h1939, &h2939
59814 DATA &h1A39, &h1C39, &h3E39, &h1739, &h3C39, &h1B39
59815 DATA &h3B39, &h0439, &h2100, &h2000, &h0939, &h0E39
59816 DATA &h0300, &h2B00, &h3F00, &h3D00, &h2A00, &h1B00
59817 DATA &h2B00, &h2C00, &h2D00, &h1D00, &h2E00, &h2F00
59818 DATA &h3000, &h4100, &h4000, &h1E00, &h1F00, &h1600
59819 DATA &h1900, &h2900, &h1A00, &h1C00, &h3E00, &h1700
59820 DATA &h3C00, &h1B00, &h3B00, &h3300, &h2139, &h3339
59821 DATA &h0339, &h1000, &h0000, &h0000, &h0000, &h0000
59822 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59823 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59824 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59825 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59826 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59827 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59828 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59829 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59830 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59831 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59832 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59833 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59834 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59835 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59836 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59837 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59838 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
```

```
59839 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59840 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59841 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59842 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59843 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59844 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59845 DATA &h0000, &h2200, &h4D00, &h4E00, &h4F00, &h5000
59846 DATA &h5100, &h5200, &h5300, &h5400, &h5500, &h5600
59847 DATA &h0000, &h0000, &h0000, &h0000, &h5700, &h5800
59848 DATA &h464B, &h474B, &h1100, &h1200, &h114B, &h124B
59849 DATA &h234B, &h0000, &h0000, &h0000, &h0000, &h0000
59850 DATA &h3B4B, &h3C4B, &h3D4B, &h3E4B, &h3F4B, &h404B
59851 DATA &h414B, &h0000, &h0000, &h0000, &h0000, &h0000
59852 DATA &h0000, &h0000, &h0000, &h0100, &h024B, &h1300
59853 DATA &h144B, &h2500, &h2600, &h3700, &h3800, &h3400
59854 DATA &h4800, &h0000, &h0000, &h224B, &h3500, &h2700
59855 DATA &h0000, &h4600, &h0000, &h4700, &h0000, &h0000
59856 DATA &h3600, &h3900, &h2300, &h2400, &h0000, &h0000
59857 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59858 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59859 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59860 DATA &h014B, &h0200, &h134B, &h1400, &h254B, &h264B
59861 DATA &h374B, &h384B, &h0000, &h484B, &h0000, &h464B
59862 DATA &h474B, &h0000, &h0000, &h5900, &h044B, &h054B
59863 DATA &h064B, &h074B, &h084B, &h094B, &h0A4B, &h0B4B
59864 DATA &h0C4B, &h0D4B, &h0E4B, &h0F4B, &h0000, &h0000
59865 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59866 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59867 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59868 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59869 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59870 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59871 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59872 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59873 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59874 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59875 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59876 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59877 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59878 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59879 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59880 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59881 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59882 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59883 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59884 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59885 DATA &h0000, &h0000
59890 DATA &H0000                                          : '*SCANCODE
59900 DATA &h0050, &h0051, &h003D, &h0021, &h0022, &h0023
59901 DATA &h0024, &h0025, &h0026, &h0027, &h0028, &h0029
59902 DATA &h0020, &h0030, &h0011, &h0031, &h005F, &h005E
59903 DATA &h0052, &h0053, &h0036, &h0070, &h0076, &h0064
59904 DATA &h0071, &h0073, &h0078, &h0074, &h0068, &h006E
59905 DATA &h006F, &h001B, &h0015, &h0035, &h000C, &h000D
59906 DATA &h0054, &h0055, &h00CC, &h0060, &h0072, &h0063
59907 DATA &h0065, &h0066, &h0067, &h0069, &h006A, &h006B
59908 DATA &h007E, &h0012, &h000F, &h0008, &h000E, &h0013
59909 DATA &h0056, &h0057, &h00CD, &h0009, &h0079, &h0077
59910 DATA &h0062, &h0075, &h0061, &h006D, &h006C, &h0033
59911 DATA &h0032, &h0014, &h00CE, &h0016, &h001A, &h0034
59912 DATA &h0034, &h0010, &h00CF, &h0018, &h0040, &h0041
59913 DATA &h0042, &h0043, &h0044, &h0045, &h0046, &h0047
59914 DATA &h0048, &h0049, &h004A, &h004B, &h0000, &h0000
59915 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59916 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59917 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59918 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59919 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59920 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59921 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59922 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59923 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59924 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59925 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
```

```
59926 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59927 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59928 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59929 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59930 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59931 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59932 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59933 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59934 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59935 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59936 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59937 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59938 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59939 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59940 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59941 DATA &h0000, &h0000, &h0000, &h0000, &h0000, &h0000
59942 DATA &h0000, &h0000, &h0000
```

```
100 DIM I.TAB%(1279),I.BUF%(1920)                    '*Dimension IRMA's tables
102 GOSUB 50000                                      '*Initialize IRMA's variables
50000 REM Initialize IRMA interface variables        '!INIT - IRMASUBS Rev 1.05
50002 RESTORE 50036                                  ' Point to initial values
50004 READ I.CRD%,I.CWR%,I.CAC%,I.CCL%,I.CKY%        ' Load command numbers
50006 READ I.CSP%,I.CXP%,I.CMD%,I.CTA%,I.CIM%
50008 READ I.MAX%,I.MTG%,I.MKY%,I.MXX%,I.MPR%       ' Main status masks
50010 READ I.MCC%,I.MDI%,I.MCM%
50012 READ I.MXX%,I.MPO%,I.MAL%,I.MDD%,I.MCI%       ' Aux status masks
50014 READ I.MRC%,I.MBC%,I.MCK%
50016 READ I.RG0%,I.RG1%,I.RG2%,I.RG3%               ' Communication registers
50018 READ I.RST%,I.RAK%,I.RAF%
50020 READ I.MAT%,I.MBS%                             ' Handshake masks
50022 READ I.VER%,I.VST%,I.VRO%,I.VCL%,I.VMS%        ' General variables
50024 READ I.VVL%,I.VFG%,I.VST$,I.VFL%,I.VT0%
50026 READ I.VT1%,I.VT2%,I.VT3%,I.VS0$,I.VS1$
50028 READ I.VAX%,I.VS2$,I.VS3$,I.VS4$,I.VS5$
50030 READ I.VS6$,I.VS7$,I.VS8$,I.VS9$,I.VT4%
50032 READ I.VT5%,I.VCB%,I.VCE%,I.VT8%,I.VT9%
50034 READ I.VOO%,I.VRR%,I.VFS%,I.VTO%
50036 DATA 0,1,2,3,4,5,6,7,8,9
50038 DATA 128,64,32,16,8,4,2,1
50040 DATA 128,64,32,16,8,4,2,1
50042 DATA &H220,&H221,&H222,&H223
50044 DATA &H226,&H227,&H227
50046 DATA 128,64
50048 DATA 0,0,0,0,0,0,0,"",0,0,0,0,0,"","",""
50050 DATA 0,"","","","","","","","","",0
50052 DATA 0,0,0,0,0,0,0,0,0,5
50058 DEF SEG
50060 BLOAD "IRMATABS.OVR",VARPTR(I.TAB%(0))
50062 RETURN
50100 REM Power on reset simulation                  '*XPOR ( I.VER% )
50102 I.VER%=0                                       ' Reset error flag
50104 OUT I.RG0%,I.CXP%                              ' Set commmand in place
50106 GOSUB 58000                                    ' Start & wait for slave
50108 RETURN
50200 REM Get slave status                           '*GSTAT ( I.VST%, I.VER% )
50202 I.VER%=0                                       ' Reset error flag
50204 OUT I.RG0%,I.CAC%                              ' Get aux status & cursor
50206 GOSUB 58000                                    ' Start & wait for slave
50208 I.VAX%=INP(I.RG3%)                             ' Get aux
50210 I.VST%=INP(I.RG0%)                             ' Get main
50212 RETURN                                         ' Exit!
50300 REM Reset slave status bits                    '*RSTAT ( I.VST%, I.VER% )
50302 I.VER%=0                                       ' Reset error flag
```

```
50304 OUT I.RG0%,I.CCL%                          ' Clear status command
50306 OUT I.RG3%,I.VST%                          ' Status bits to clear
50308 GOSUB 58000                                ' Start & wait for slave
50310 RETURN                                     ' Exit!
50400 REM Set trigger event & address            '*STDNM ( I.VRO%, I.VCL% )
50401 I.VER%=0                                   '          I.VMS%, I.VVL%
50402 GOSUB 58200                                ' Check row and column values
50403 IF I.VER%<>0 THEN RETURN                   ' Error if bad input
50404 OUT I.RG0%,I.CTA%                          '          I.VER%          )
50406 I.VT0%=(I.VCL%-1)+(I.VRO%*80)              ' Compute address
50408 OUT I.RG2%,I.VT0%\&H100                    ' High part of address
50410 OUT I.RG1%,I.VT0% AND &HFF                 ' Low part of address
50411 GOSUB 58000                                ' Start slave
50412 IF ((I.VVL% OR I.VMS% ) AND &HFF00)<>0 THEN I.VER%=4 : RETURN ' Bad byte
50417 IF I.VER%<>0 THEN RETURN                   ' Give up if dead slave
50418 OUT I.RG0%,I.CMD%                          ' Setup mask & data
50419 OUT I.RG1%,I.VVL%                          ' Data
50420 OUT I.RG2%,I.VMS%                          ' Mask
50422 GOSUB 58000                                ' Set trigger.
50424 RETURN                                     ' Exit!
50500 REM Wait for trigger event                 '*WTRIG ( I.VT0%, I.VER% )
50502 I.VER%=0                                   ' Reset error flag
50504 I.VS0$=TIME$                               ' Set the stopwatch
50506 IF (INP(I.RG0%) AND I.MTG%)<>0 THEN RETURN' All done!
50508 I.VER%=10                                  ' Potential timeout.
50510 IF I.VT0%<0 THEN RETURN                    ' Time has run out.
50512 IF I.VS0$=TIME$ THEN 50502                 ' Still time, try again
50514 I.VT0%=I.VT0%-1                            ' Drop a grain of sand.
50516 GOTO 50502                                 ' Do it some more.
50600 REM Send keystrokes from I.VST$            '*KEYS ( I.VST$, I.VER% ))
50602 I.VER%=0                                   ' Reset error flag
50604 I.VT0%=LEN(I.VST$)                         ' Count of remaining chars
50606 I.VT1%=1                                   ' Current pointer
50608 WHILE I.VT0%>0                             ' Only until none remain...
50610   I.VT2%=ASC(MID$(I.VST$,I.VT1%,1))        ' Get character value
50612   IF I.VT2%>0 THEN 50620                   ' Not an extended code...
50614     I.VT0%=I.VT0%-1 : I.VT1%=I.VT1%+1      ' Get set to eat next char
50616     IF I.VT0%<1 THEN 50658                 ' EXIT if partial char
50618     I.VT2%=ASC(MID$(I.VST$,I.VT1%,1))+256  ' Offset into extended table
50620   I.VT2%=I.TAB%(I.VT2%+&H200)              ' Look up key codes
50622   IF (I.VT2% AND &HFF)=0 THEN 50634        ' Skip shift key
50624     I.VT3%=I.TAB%((I.VT2% AND &HFF)+&H400)' Get scan code of shift
50626     I.VT3%=I.VT3% AND &H7F                 ' Strip up/dn control bit
50628     GOSUB 58100                            ' Transmit scan code
50630     IF I.VER%=0 THEN 50634                 ' Skip error exit
50632       I.VT0%=-1 : GOTO 50658               ' Error on key attempt ABORT!
50634   IF (I.VT2% AND &HFF00)=0 THEN 50644      ' Handle possible lone shift
50636     I.VT3%=I.TAB%((I.VT2%\&H100)+&H400)    ' Get scan code
50638     GOSUB 58100                            ' Transmit it.
50640     IF I.VER%=0 THEN 50644                 ' Skip error exit
50642       I.VT0%=-1 : GOTO 50658               ' PUNT.
50644   IF (I.VT2% AND &HFF)=0 THEN 50654        ' Skip shift key
50646     I.VT3%=I.TAB%((I.VT2% AND &HFF)+&H400)' Get scan code of shift
50648     GOSUB 58100                            ' Transmit scan code
50650     IF I.VER%=0 THEN 50654                 ' Skip error exit
50652       I.VT0%=-1 : GOTO 50658               ' Error on key attempt ABORT!
50654   I.VT0%=I.VT0%-1                          ' One less character to send
50656   I.VT1%=I.VT1%+1                          ' Point to next character
50658 WEND                                       ' Do until done or error
50660 RETURN
50700 REM Find field I.VFL%                      '*FIND ( I.VFL%, I.VCL% ))
50701 I.VER%=0                                   '          I.VRO%, I.VER% )
50703 IF (I.VFL%<1) OR (I.VFL%>1920) THEN I.VER%=7 : RETURN
50704 I.VCE%=80 : I.VCB%=80 : I.BUF%(0)=&HC0     ' Start at upper left
50706 OUT I.RG0%,I.CRD%                          ' Preset the read command
50708 IF I.VFL%=1 THEN 50738                     ' Default screen condition
50710 I.VT1%=1                                   ' Current field number
50712 OUT I.RG1%,I.VCE% AND &HFF                 ' Low order address
50714 OUT I.RG2%,I.VCE%\&H100                    ' High order address
50716 GOSUB 58000                                ' Start slave & wait
50718 IF I.VER%<>0 THEN RETURN                   ' PUNT if read error.
50720 I.VT2%=INP(I.RG3%)                         ' Get data
50722 IF I.VT2%<&HC0 THEN 50730                  ' Skip if not attribute
50724   IF (I.VT2% AND &H20)>0 THEN 50730        ' Skip if protected
```

```
50725      I.VCB%=I.VCE%                                  ' Save start position
50726      I.VT1%=I.VT1%+1                                ' We've found the next field!
50727      I.BUF%(0)=I.VT2%                               ' Init buffer attribute
50728      IF I.VT1%=I.VFL% THEN 50736                    ' Exit the search loop
50730 I.VCE%=I.VCE%+1                                    ' Try next location
50732 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN             ' Field not found anywhere!
50734 GOTO 50712                                         ' Read the next character
50736 I.VCE%=I.VCE%+1                                    ' Point to first char of field
50738 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN             ' Field at last col/row
50740 I.VRO%=I.VCE%\80                                   ' Make the row number
50742 I.VCL%=(I.VCE% MOD 80)+1                           ' And column number
50744 I.VFS%=0                                           ' Length of field
50746 OUT I.RG1%,I.VCE% AND &HFF                         ' Low order address
50748 OUT I.RG2%,I.VCE%\&H100                            ' High order address
50750 GOSUB 58000                                        ' Start read operation
50752 IF I.VER%<>0 THEN RETURN                           ' Quit if slave is dead
50754 I.VT2%=INP(I.RG3%)                                 ' Get data
50756 IF I.VT2%>=&HC0 THEN RETURN                        ' Next attribute found
50758 I.VFS%=I.VFS%+1                                    ' Count the chars in field
50760 I.VCE%=I.VCE%+1                                    ' Point to next char
50762 IF I.VCE%>=2000 THEN RETURN                        ' Ran off end of screen!
50764 GOTO 50746                                         ' Continue eating chars
50800 REM Find NEXT field I.VFL%+1                       '*FNEXT ( I.VFL%, I.VCL%,)
50801 I.VER%=0                                           '       ( I.VRO%, I.VER% )
50802 I.VT1%=I.VFL% : I.VFL%=I.VFL%+1                   ' Next field
50804 IF I.VCE%<80 THEN I.VER%=11 : RETURN               ' Illegal cursor value
50806 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN             ' No next field.
50808 OUT I.RG0%,I.CRD%                                  ' Set read command in place
50810 OUT I.RG1%,I.VCE% AND &HFF                         ' Low order address
50812 OUT I.RG2%,I.VCE%\&H100                            ' High order address
50814 GOSUB 58000                                        ' Get data at current position
50816 IF I.VER%>0 THEN RETURN                            ' Give up if slave dead
50820 I.VT9%=INP(I.RG3%)                                 ' This should be attribute
50822 IF I.VT9%<&HC0 THEN I.VER%=11 : RETURN             ' Invalid cursor position
50830 GOTO 50712                                         ' Enter general FIND code
50900 REM Read field                                     '*RDFLD ( I.VER% )I.VCL%,)
50902 I.VER%=0 : I.VT2%=0                                ' Reset error flag & buffer ptr
50904 IF I.VCB%=80 THEN 50924                            ' Special case of upper corner
50906 IF I.VCB%<80 THEN I.VER%=11                        ' Invalid start address
50908 IF I.VCB%>=2000 THEN I.VER%=11                     ' Beyond end of screen
50910 OUT I.RG0%,I.CRD%                                  ' Set read command
50912 OUT I.RG1%,I.VCB% AND &HFF                         ' Low order address
50914 OUT I.RG2%,I.VCB%\&H100                            ' High order address
50916 GOSUB 58000                                        ' Read the leading attribute
50918 I.VT3%=INP(I.RG3%)                                 ' Data word
50920 IF (I.VT3% AND &HE0)<>&HC0 THEN I.VER%=5  : RETURN ' Bad field type
50922 GOTO 50930
50924 I.VT2%=1                                           ' Scan and eat field
50926 I.BUF%(0)=&HC0                                     ' Increment buffer pointer
50930 OUT I.RG0%,I.CRD%                                  ' Fake attribute
50934 WHILE I.VT2%<=I.VFS%                               ' Setup the read command
50936   I.VT9%=I.VT2%+I.VCB%                             ' Until out of characters
50938   IF I.VT9%>=2000 THEN I.VER%=11 : GOTO 50950 ' bad field specs
50940   OUT I.RG1%,I.VT9% AND &HFF                       ' Offset to character
50942   OUT I.RG2%,I.VT9%\&H100                          ' Set low address
50946   GOSUB 58000                                      ' High address
50948   IF I.VER%=0 THEN 50960                           ' Xecute read operation
50950     I.VT2%=9999                                    ' Abort if slave dead
50952     GOTO 50970                                     ' Exit loop
50960   I.BUF%(I.VT2%)=CVI(CHR$(INP(I.RG3%))+CHR$(INP(I.RG2%)))
50962   I.VT2%=I.VT2%+1                                  ' Point to next location
50970 WEND
50972 RETURN                                             ' All done
51000 REM Write field                                    '*WRFLD ( I.VER% )I.VCL%,)
51002 I.VER%=0                                           ' Reset error flag
51004 IF I.VCB%<80 THEN I.VER%=11                        ' Illegal address
51006 IF I.VCB%>=2000 THEN I.VER%=11                     ' Too large
51008 IF I.VCB%+I.VT1%>2000 THEN I.VER%=12               ' Field too long for screen
51010 IF I.VER%<>0 THEN RETURN                           ' Quit if bad parameters
51012 OUT I.RG0%,I.CRD%                                  ' Read attribute from screen
51014 OUT I.RG1%,I.VCB% AND &HFF                         ' Low address
51016 OUT I.RG2%,I.VCB%\&H100                            ' High address
51018 GOSUB 58000                                        ' Xecute!
51020 IF I.VER%<>0 THEN RETURN                           ' No good news!
```

```
51022 IF (INP(I.RG3%) AND &HFE)<>(I.BUF%(0) AND &HFE) THEN I.VER%=13 : RETURN
51024 I.BUF%(0)=I.BUF%(0) OR 1                    ' Set the MDT flag
51026 I.VT2%=0                                   ' First byte of buffer to write
51028 IF (I.BUF%(0) AND &H10)=0 THEN 51052        ' Check for numeric only
51030   I.VT2%=1                                  ' First buffer location to chk
51032   WHILE I.VT2%<=I.VT1%                      ' Do until all chars done
51034     I.VT0%=I.BUF%(I.VT2%) AND &HFF          ' Mask off any previous EAB
51036     IF I.VT0%>=&H20 THEN IF I.VT0%<=&H29 THEN 51048 ' 0-9 is Ok
51038       IF I.VT0%=&H31 THEN 51048             ' Minus Ok
51039       IF I.VT0%=&H32 THEN 51048             ' Period Ok
51040       IF I.VT0%=&H35 THEN 51048             ' Plus Ok
51042     I.VER%=6 : I.VT2%=9999                  ' "Non-numeric" char
51048     I.VT2%=I.VT2%+1                         ' Next character
51050   WEND                                      ' Loop
51052 I.VT2%=0                                   ' Start from the beginning
51054 OUT I.RG0%,I.CWR%                          ' Set write command
51056 WHILE I.VT2%<=I.VFS%                       ' Until all characters written
51058   I.VT0%=I.VCB%+I.VT2%                     ' Compute address
51060   OUT I.RG1%,I.VT0% AND &HFF               ' Low address
51062   OUT I.RG2%,I.VT0%\&H100                  ' High address
51064   OUT I.RG3%,I.BUF%(I.VT2%) AND &HFF       ' Data w/o EAB
51066   GOSUB 58000                              ' Make slave do it.
51068   IF I.VER%<>0 THEN I.VT2%=9999            ' Force exit if slave DOA
51070   I.VT2%=I.VT2%+1                          ' Next location to write
51072 WEND                                       ' Do it again
51074 RETURN                                     ' All done!
51100 REM Get string from buffer                 '*GSTR( I.VST$, I.VT1%
51105 I.VER%=0 : I.VT2%=0                        '       I.VT0%          )
51108 IF I.VOO%>1920 THEN I.VER%=12 : RETURN     ' Offset too large
51110 IF I.VOO%<0 THEN I.VER%=12 : RETURN        ' Offset negative
51112 I.VST$=""                                  ' Clear string
51114 WHILE (I.VOO%<=I.VFS%) AND (I.VT2%=0)      '
51116   I.VST$=I.VST$+CHR$(I.TAB%((I.BUF%(I.VOO%) AND &HFF)+&H100))
51118   I.VOO%=I.VOO%+1                          ' Point to next char
51120   IF LEN(I.VST$)=254 THEN I.VT2%=1         ' Overrun
51122 WEND
51126 RETURN
51200 REM Put string in buffer                   '*PSTR( I.VST$, I.VT1%
51202 I.VER%=0                                   '       I.VT0%          )
51204 IF I.VOO%+LEN(I.VST$)-1>I.VFS% THEN I.VER%=12 : RETURN ' Too long
51206 I.VT3%=LEN(I.VST$) : IF I.VT3%=0 THEN RETURN ' Zero strings easy!
51208 I.VT2%=0                                   ' Offset into buffer
51210 WHILE I.VT2%<I.VT3%                        ' While still characters
51212   I.BUF%(I.VOO%+I.VT2%)=I.TAB%(ASC(MID$(I.VST$,I.VT2%+1,1))+&H0)
51214   I.VT2%=I.VT2%+1                          ' Move pointer
51216 WEND
51218 I.VOO%=I.VOO%+I.VT2%                       ' Pointer for next
51220 RETURN                                     ' Transfer complete
51300 REM Read screen absolute                   '*ABSRD ( I.VER%, I.VST$,
51302 I.VER%=0                                   '         I.VS0$, I.VRO%,
51304 I.VT0%=I.VRO%*80+I.VCL%-1                  '         I.VCL%, I.VRR% )
51306 IF I.VT0%<0 THEN I.VER%=11 : RETURN        ' Invalid cursor address
51308 IF I.VT0%>=2000 THEN I.VER%=11 : RETURN    '
51310 IF I.VT0%+I.VRR%>2000 THEN I.VER%=12 : RETURN ' Can't read that much
51312 I.VT1%=0 : I.VST$="" : I.VS0$=""           ' Offset from start
51314 OUT I.RG0%,I.CRD% : I.VT2%=I.VT0%+I.VRR%   ' Set a read commnad
51316 WHILE I.VT0%<I.VT2%                        ' Until all characters read
51318   OUT I.RG1%,I.VT0% AND &HFF               ' Low order address
51320   OUT I.RG2%,I.VT0%\&H100                  ' High order address
51322   GOSUB 58000                              ' Execute read operation
51324   IF I.VER%<>0 THEN I.VT0%=9999            ' Force loop exit
51326   I.VST$=I.VST$+CHR$(I.TAB%(INP(I.RG3%)+&H100)) ' Get char & convert
51328   I.VS0$=I.VS0$+CHR$(INP(I.RG2%))          ' Unmodified EAB
51340   I.VT0%=I.VT0%+1                          ' Next character
51342 WEND
51344 RETURN
51400 REM Read 3278 cursor position              '*CPOS( I.VRO%, I.VCL%,
51402 I.VER%=0                                   '       I.VER%          )
51406 OUT I.RG0%,I.CAC%                          ' Status & cursor read
51408 GOSUB 58000                                ' Start the slave
51410 I.VT0%=(INP(I.RG2%)*&H100)+INP(I.RG1%)     ' Get absolute address
51411 IF I.VER%<>0 THEN RETURN                   ' Dead slave, quit action
51412 I.VRO%=I.VT0%\80                           ' Compute row
51414 I.VCL%=(I.VT0% MOD 80)+1                   ' And column
```

```
51416 RETURN
51500 REM Read absolute memory position          '*Read memory
51502 OUT I.RG0%,I.CRD%                          ' Set read command
51504 OUT I.RG1%,I.VOO% AND &HFF                 ' Low address
51506 OUT I.RG2%,I.VOO%\&H100                    ' High address
51508 GOSUB 58000                                ' Start & wait for s?
51520 IF I.VER%<>0 THEN RETURN                   ' No good news!
51522 I.VVL%=INP(I.RG3%)                         ' Get the data
51524 RETURN                                     ' And egzit!
58000 REM Start & wait for slave'!Start & wait for slave
58002 I.VS9$=TIME$ : I.VT9%=0                    ' A simulated stopwatch
58004 OUT I.RST%,0                               ' Start the slave
58006 I.VT8% = INP( I.RAF% ) AND I.MBS%          ' Get slave busy bit
58008 IF I.VT8% = 0 THEN RETURN                  ' Return with command complete
58010 IF I.VS9$=TIME$ THEN 58006                 ' Loop until clock ticks
58012 I.VS9$=TIME$ : I.VT9%=I.VT9%+1             ' Three times
58014 IF I.VT9%<3 THEN 58006                     '
58016 I.VER%=1                                   ' Slave timeout
58018 RETURN                                     '
58100 REM Send key scan code from I.VT3%'*Send key scan code
58102 I.VS9$=TIME$ : I.VT9%=0                    ' Make stopwatch
58104 IF (INP(I.RG0%) AND I.MKY%)>0 THEN 58114   ' Key buffer is ready
58106 IF I.VS9$=TIME$ THEN 58104                 ' Loop until clock tick
58108 I.VT9%=I.VT9%+1 : IF I.VT9%<4 THEN 58104   ' Time has not run out
58110 I.VER%=9                                   ' Keystroke timeout
58112 RETURN
58114 IF I.VT3%=0 THEN I.VER%=15 : RETURN        ' Invalid scan code
58115 OUT I.RG0%,I.CKY%                          ' Keystroke command
58116 OUT I.RG3%,I.VT3%                          ' Scan code
58118 GOSUB 58000                                ' Fire up the slave
58120 RETURN                                     ' And we're done!
58200 REM - Verify Row and Column'*Verify ROW/COL
58206 IF (I.VRO%<0) OR (I.VRO%>24) THEN I.VER%=2 : RETURN
58208 IF (I.VCL%<1) OR (I.VCL%>80) THEN I.VER%=3 : RETURN
58210 RETURN
```

```
10 CLS : REM IRMASUBS SAMPLES 26 July 83
20 PRINT "IRMASUBS Programming Guide and Samples - IRMASUBS SAMPLE 1.20"
22 PRINT
24 PRINT "This program is intended as a programming and study guide only."
26 PRINT "      Do not attempt to continue execution of this program."
28 PRINT
30 PRINT "This program, if executed in interpreted BASICA requires a"
32 PRINT "specific IBM test screen initially and more than ONE HOUR to"
34 PRINT "complete execution."
36 PRINT
40 END
50 DEFINT A-Z
52 PRINT "IRMA BASIC SUBROUTINES DEMONSTRATION - IRMASUBS DEMO 1.01
54 PRINT : PRINT "Initializing IRMASUBS variables & tables
100 DIM I.TAB%(1279),I.BUF%(1920)                '*Dimension IRMA's tables
102 GOSUB 50000                                  '*Initialize IRMA's variables
110 PRINT
200 REM Field Report                             '*Field Report
202 I.VFL%=1                                     ' Start with the first field
204 PRINT "Field Row Column Length Contents"     ' Print field info header
206   FH$=" ####  ##    ##     ####  &"          ' Print using definition
210 GOSUB 50700                                  ' Find the first field
212 WHILE I.VER%=0                               ' Until no fields found
214   GOSUB 50900                                ' Read field contents
216   I.VOO%=1                                   ' Get string from buffer start
218   GOSUB 51100                                ' Read string I.VST$
220   I.VST$=LEFT$(I.VST$,40)                    ' Limit the length to screen
222   PRINT USING FH$;I.VFL%,I.VRO%,I.VCL%,I.VFS%,I.VST$
230   GOSUB 50800                                ' Find next field
```

```
240 WEND
300 REM Modify a few fields                          '*Modify field
302 I.VFL%=2                                         ' Second screen field
310 GOSUB 50700                                      ' Find the first field
320 I.VST$="Two"                                     ' A simple string
322 I.VOO%=1                                         ' Beginning of data area
324 GOSUB 51200                                      ' Put string in buffer
326 GOSUB 51000                                      ' Write the field
328 IF I.VER%<>0 THEN PRINT "Error: ";I.VER%         ' Report possible error
330 GOSUB 50800                                      ' Point to next field
332 I.VST$="Three"                                   ' More nice data
334 I.VOO%=1                                         ' Buffer start
336 GOSUB 51200                                      ' Put new string in buffer
338 IF I.VFS%>LEN(I.VST$) THEN I.VFS%=LEN(I.VST$) ' Shorten write length
340 GOSUB 51000                                      ' Write the field
345 STOP
400 REM Display part of the screen                   '*Display
402 I.VCL%=1 : I.VRR%=40                             ' Column one
404 FOR I.VRO%=1 TO 5                                ' Top five lines
406   GOSUB 51300                                    ' Read a short line
410   PRINT I.VST$                                   ' Print the line
412 NEXT I.VRO%                                      ' Again, with feeling
500 REM General status info                          '*Status info
502 GOSUB 51400                                      ' Read cursor postion
504 PRINT "Buffer pointer ROW:";I.VRO%;"  COLUMN:";I.VCL%
510 GOSUB 50200                                      ' Get slave status
520 GOSUB 900                                        ' Display status
600 REM Send some keystrokes                         '*Keystrokes
602 I.VST$=CHR$(0)+CHR$(15)+"AAA"                    ' Back tab character w/ string
604 GOSUB 50600                                      ' Send keystrokes
612 I.VST$=CHR$(9)+"bbb"                             ' Forward tab
614 GOSUB 50600                                      ' Send keystrokes
620 STOP
700 REM Clear any status                             '*Clear status
702 GOSUB 50200                                      ' Read current status
706 GOSUB 900                                        ' Display current status
710 GOSUB 50300                                      ' Clear all set bits
712 GOSUB 50200                                      ' Get the status again
714 GOSUB 900                                        ' Display the new status
800 REM Trigger stuffs                               '*Trigger tests
802 I.VRO%=1 : I.VCL%=1                              ' Upper left corner
804 I.VMS%=0 : I.VVL%=0                              ' Any change is a trigger
806 GOSUB 50400                                      ' Set trigger data & addr
808 PRINT "Waiting for any change in row 1, column 1"
820 WHILE LEN(INKEY$)=0                              ' Until a key is pressed
822   I.VTO%=2                                       ' 2 second timeout
824   GOSUB 50500                                    ' Wait for trigger event
826   IF I.VTO%<0 THEN PRINT "."; : GOTO 840
830   PRINT "Trigger! "
836   GOTO 842
840 WEND
842 I.VST%=64                                        ' Clear trigger bit
844 GOSUB 50300                                      '
848 PRINT "Waiting for an upper case A in row 1, column 1"
850 I.VMS%=255 : I.VVL%=160                          ' Specific Upper case A
852 GOSUB 50400                                      ' Set the trigger again
860 WHILE LEN(INKEY$)=0                              ' Until a key is pressed
862   I.VTO%=2                                       ' 2 second timeout
864   GOSUB 50500                                    ' Wait for trigger event
866   IF I.VTO%<0 THEN PRINT "."; : GOTO 870
867   PRINT "Trigger! "
868   GOTO 872
870 WEND
872 I.VST%=64                                        ' Clear trigger bit
874 GOSUB 50300                                      '
899 END
900 REM Display status words                         '!Status display
912 PRINT
918 PRINT "Main status word:"
920 IF (I.VST% AND I.MAX%) THEN PRINT "  Aux status change"
922 IF (I.VST% AND I.MTG%) THEN PRINT "  Trigger occurred"
924 IF (I.VST% AND I.MKY%) THEN PRINT "  Key buffer empty"
926 IF (I.VST% AND I.MPR%) THEN PRINT "  Controller issued reset"
928 IF (I.VST% AND I.MCC%) THEN PRINT "  Last command complete"
```

```
930 IF (I.VST% AND I.MDI%) THEN PRINT "  Buffer dirty (modified)"
932 IF (I.VST% AND I.MCM%) THEN PRINT "  Buffer pointer moved"
938 PRINT "Aux status word:"
940 IF (I.VAX% AND I.MPO%) THEN PRINT "  Poll occurred
942 IF (I.VAX% AND I.MAL%) THEN PRINT "  Alarm requested"
944 IF (I.VAX% AND I.MDD%) THEN PRINT "  Display disabled (Inhibited)"
946 IF (I.VAX% AND I.MCI%) THEN PRINT "  Cursor inhibited"
948 IF (I.VAX% AND I.MRC%) THEN PRINT "  Reverse video cursor"
950 IF (I.VAX% AND I.MBC%) THEN PRINT "  Blinking cursor"
952 IF (I.VAX% AND I.MCK%) THEN PRINT "  Keyboard clicker enabled"
980 RETURN
50000 REM Initialize IRMA interface variables    '!INIT - IRMASUBS Rev 1.01
50002 RESTORE 50036                              ' Point to initial values
50004 READ I.CRD%,I.CWR%,I.CAC%,I.CCL%,I.CKY%    ' Load command numbers
50006 READ I.CSP%,I.CXP%,I.CMD%,I.CTA%,I.CIM%
50008 READ I.MAX%,I.MTG%,I.MKY%,I.MXX%,I.MPR%    ' Main status masks
50010 READ I.MCC%,I.MDI%,I.MCM%
50012 READ I.MXX%,I.MPO%,I.MAL%,I.MDD%,I.MCI%    ' Aux status masks
50014 READ I.MRC%,I.MBC%,I.MCK%
50016 READ I.RG0%,I.RG1%,I.RG2%,I.RG3%           ' Communication registers
50018 READ I.RST%,I.RAK%,I.RAF%
50020 READ I.MAT%,I.MBS%                         ' Handshake masks
50022 READ I.VER%,I.VST%,I.VRO%,I.VCL%,I.VMS%    ' General variables
50024 READ I.VVL%,I.VFG%,I.VST$,I.VFL%,I.VT0%
50026 READ I.VT1%,I.VT2%,I.VT3%,I.VS0$,I.VS1$
50028 READ I.VAX%,I.VS2$,I.VS3$,I.VS4$,I.VS5$
50030 READ I.VS6$,I.VS7$,I.VS8$,I.VS9$,I.VT4%
50032 READ I.VT5%,I.VCB%,I.VCE%,I.VT8%,I.VT9%
50034 READ I.VOO%,I.VRR%,I.VFS%,I.VTO%
50036 DATA 0,1,2,3,4,5,6,7,8,9
50038 DATA 128,64,32,16,8,4,2,1
50040 DATA 128,64,32,16,8,4,2,1
50042 DATA &H220,&H221,&H222,&H223
50044 DATA &H226,&H227,&H227
50046 DATA 128,64
50048 DATA 0,0,0,0,0,0,0,"",0,0,0,0,0,"",""
50050 DATA 0,"","","","","","","","",0
50052 DATA 0,0,0,0,0,0,0,0,5
50058 DEF SEG
50060 BLOAD "IRMATABS.OVR",VARPTR(I.TAB%(0))
50062 RETURN
50100 REM Power on reset simulation              '*XPOR ( I.VER% )
50102 I.VER%=0                                   ' Reset error flag
50104 OUT I.RG0%,I.CXP%                          ' Set commmand in place
50106 GOSUB 58000                                ' Start & wait for slave
50108 RETURN
50200 REM Get slave status                       '*GSTAT ( I.VST%, I.VER% )
50202 I.VER%=0                                   ' Reset error flag
50204 OUT I.RG0%,I.CAC%                          ' Get aux status & cursor
50206 GOSUB 58000                                ' Start & wait for slave
50208 I.VAX%=INP(I.RG3%)                         ' Get aux
50210 I.VST%=INP(I.RG0%)                         ' Get main
50212 RETURN                                     ' Exit!
50300 REM Reset slave status bits                '*RSTAT ( I.VST%, I.VER% )
50302 I.VER%=0                                   ' Reset error flag
50304 OUT I.RG0%,I.CCL%                          ' Clear status command
50306 OUT I.RG3%,I.VST%                          ' Status bits to clear
50308 GOSUB 58000                                ' Start & wait for slave
50310 RETURN                                     ' Exit!
50400 REM Set trigger event & address            '*STDNM ( I.VRO%, I.VCL% )
50401 I.VER%=0                                   '         I.VMS%, I.VVL%
50402 GOSUB 58200                                ' Check row and column values
50403 IF I.VER%<>0 THEN RETURN                   ' Error if bad input
50404 OUT I.RG0%,I.CTA%                          '         I.VER%          )
50406 I.VT0%=(I.VCL%-1)+(I.VRO%*80)              ' Compute address
50408 OUT I.RG2%,I.VT0%\&H100                    ' High part of address
50410 OUT I.RG1%,I.VT0% AND &HFF                 ' Low part of address
50411 GOSUB 58000                                ' Start slave
50412 IF ((I.VVL% OR I.VMS% ) AND &HFF00)<>0 THEN I.VER%=4 : RETURN ' Bad byte
50417 IF I.VER%<>0 THEN RETURN                   ' Give up if dead slave
50418 OUT I.RG0%,I.CMD%                          ' Setup mask & data
50419 OUT I.RG1%,I.VVL%                          ' Data
50420 OUT I.RG2%,I.VMS%                          ' Mask
50422 GOSUB 58000                                ' Set trigger.
```

```
50424 RETURN                                          ' Exit!
50500 REM Wait for trigger event                      '*WTRIG ( I.VTO%, I.VER% )
50502 I.VER%=0                                        ' Reset error flag
50504 I.VS0$=TIME$                                    ' Set the stopwatch
50506 IF (INP(I.RG0%) AND I.MTG%)<>0 THEN RETURN      ' All done!
50508 I.VER%=10                                       ' Potential timeout.
50510 IF I.VTO%<0 THEN RETURN                         ' Time has run out.
50512 IF I.VS0$=TIME$ THEN 50502                      ' Still time, try again
50514 I.VTO%=I.VTO%-1                                 ' Drop a grain of sand.
50516 GOTO 50502                                      ' Do it some more.
50600 REM Send keystrokes from I.VST$                 '*KEYS ( I.VST$, I.VER% ))
50602 I.VER%=0                                        ' Reset error flag
50604 I.VT0%=LEN(I.VST$)                              ' Count of remaining chars
50606 I.VT1%=1                                        ' Current pointer
50608 WHILE I.VT0%>0                                  ' Only until none remain...
50610   I.VT2%=ASC(MID$(I.VST$,I.VT1%,1))             ' Get character value
50612   IF I.VT2%>0 THEN 50620                        ' Not an extended code...
50614     I.VT0%=I.VT0%-1 : I.VT1%=I.VT1%+1           ' Get set to eat next char
50616     IF I.VT0%<1 THEN 50658                      ' EXIT if partial char
50618     I.VT2%=ASC(MID$(I.VST$,I.VT1%,1))+256       ' Offset into extended table
50620   I.VT2%=I.TAB%(I.VT2%+&H200)                   ' Look up key codes
50622   IF (I.VT2% AND &HFF)=0 THEN 50634             ' Skip shift key
50624     I.VT3%=I.TAB%((I.VT2% AND &HFF)+&H400)      ' Get scan code of shift
50626     I.VT3%=I.VT3% AND &H7F                      ' Strip up/dn control bit
50628     GOSUB 58100                                 ' Transmit scan code
50630     IF I.VER%=0 THEN 50634                      ' Skip error exit
50632       I.VT0%=-1 : GOTO 50658                    ' Error on key attempt ABORT!
50634   IF (I.VT2% AND &HFF00)=0 THEN 50644           ' Handle possible lone shift
50636     I.VT3%=I.TAB%((I.VT2%\&H100)+&H400)         ' Get scan code
50638     GOSUB 58100                                 ' Transmit it.
50640     IF I.VER%=0 THEN 50644                      ' Skip error exit
50642       I.VT0%=-1 : GOTO 50658                    ' PUNT.
50644   IF (I.VT2% AND &HFF)=0 THEN 50654             ' Skip shift key
50646     I.VT3%=I.TAB%((I.VT2% AND &HFF)+&H400)      ' Get scan code of shift
50648     GOSUB 58100                                 ' Transmit scan code
50650     IF I.VER%=0 THEN 50654                      ' Skip error exit
50652       I.VT0%=-1 : GOTO 50658                    ' Error on key attempt ABORT!
50654   I.VT0%=I.VT0%-1                               ' One less character to send
50656   I.VT1%=I.VT1%+1                               ' Point to next character
50658 WEND                                            ' Do until done or error
50660 RETURN
50700 REM Find field I.VFL%                           '*FIND ( I.VFL%, I.VCL% ))
50702 I.VER%=0                                        '       I.VRO%, I.VER% )
50703 IF (I.VFL%<1) OR (I.VFL%>1920) THEN I.VER%=7 : RETURN
50704 I.VCE%=80 : I.VCB%=80 : I.BUF%(0)=&HC0          ' Start at upper left
50706 OUT I.RG0%,I.CRD%                               ' Preset the read command
50708 IF I.VFL%=1 THEN 50738                          ' Default screen condition
50710 I.VT1%=1                                        ' Current field number
50712 OUT I.RG1%,I.VCE% AND &HFF                      ' Low order address
50714 OUT I.RG2%,I.VCE%\&H100                         ' High order address
50716 GOSUB 58000                                     ' Start slave & wait
50718 IF I.VER%<>0 THEN RETURN                        ' PUNT if read error.
50720 I.VT2%=INP(I.RG3%)                              ' Get data
50722 IF I.VT2%<&HC0 THEN 50730                       ' Skip if not attribute
50724   IF (I.VT2% AND &H20)>0 THEN 50730             ' Skip if protected
50725     I.VCB%=I.VCE%                               ' Save start position
50726     I.VT1%=I.VT1%+1                             ' We've found the next field!
50727     I.BUF%(0)=I.VT2%                            ' Init buffer attribute
50728     IF I.VT1%=I.VFL% THEN 50736                 ' Exit the search loop
50730 I.VCE%=I.VCE%+1                                 ' Try next location
50732 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN          ' Field not found anywhere!
50734 GOTO 50712                                      ' Read the next character
50736 I.VCE%=I.VCE%+1                                 ' Point to first char of field
50738 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN          ' Field at last col/row
50740 I.VRO%=I.VCE%\80                                ' Make the row number
50742 I.VCL%=(I.VCE% MOD 80)+1                        ' And column number
50744 I.VFS%=0                                        ' Length of field
50746 OUT I.RG1%,I.VCE% AND &HFF                      ' Low order address
50748 OUT I.RG2%,I.VCE%\&H100                         ' High order address
50750 GOSUB 58000                                     ' Start read operation
50752 IF I.VER%<>0 THEN RETURN                        ' Quit if slave is dead
50754 I.VT2%=INP(I.RG3%)                              ' Get data
50756 IF I.VT2%>=&HC0 THEN RETURN                     ' Next attribute found
50758 I.VFS%=I.VFS%+1                                 ' Count the chars in field
```

```
50760 I.VCE%=I.VCE%+1                                   ' Point to next char
50762 IF I.VCE%>=2000 THEN RETURN                       ' Ran off end of screen!
50764 GOTO 50746                                        ' Continue eating chars
50800 REM Find NEXT field I.VFL%+1                      '*FNEXT ( I.VFL%, I.VCL%,)
50801 I.VER%=0                                          '       ( I.VRO%, I.VER% )
50802 I.VT1%=I.VFL% : I.VFL%=I.VFL%+1                  ' Next field
50804 IF I.VCE%<80 THEN I.VER%=11 : RETURN              ' Illegal cursor value
50806 IF I.VCE%>=2000 THEN I.VER%=7 : RETURN            ' No next field.
50808 OUT I.RG0%,I.CRD%                                 ' Set read command in place
50810 OUT I.RG1%,I.VCE% AND &HFF                        ' Low order address
50812 OUT I.RG2%,I.VCE%\&H100                           ' High order address
50814 GOSUB 58000                                       ' Get data at current position
50816 IF I.VER%>0 THEN RETURN                           ' Give up if slave dead
50820 I.VT9%=INP(I.RG3%)                                ' This should be attribute
50822 IF I.VT9%<&HC0 THEN I.VER%=11 : RETURN            ' Invalid cursor position
50830 GOTO 50712                                        ' Enter general FIND code
50900 REM Read field                                    '*RDFLD ( I.VER% )I.VCL%,)
50902 I.VER%=0 : I.VT2%=0                               ' Reset error flag & buffer ptr
50904 IF I.VCB%=80 THEN 50924                           ' Special case of upper corner
50906 IF I.VCB%<80 THEN I.VER%=11                       ' Invalid start address
50908 IF I.VCB%>=2000 THEN I.VER%=11                    ' Beyond end of screen
50910 OUT I.RG0%,I.CRD%                                 ' Set read command
50912 OUT I.RG1%,I.VCB% AND &HFF                        ' Low order address
50914 OUT I.RG2%,I.VCB%\&H100                           ' High order address
50916 GOSUB 58000                                       ' Read the leading attribute
50918 I.VT3%=INP(I.RG3%)                                ' Data word
50920 IF (I.VT3% AND &HE0)<>&HC0 THEN I.VER%=5  : RETURN ' Bad field type
50922 GOTO 50930                                        ' Scan and eat field
50924 I.VT2%=1                                          ' Increment buffer pointer
50926 I.BUF%(0)=&HC0                                    ' Fake attribute
50930 OUT I.RG0%,I.CRD%                                 ' Setup the read command
50934 WHILE I.VT2%<=I.VFS%                              ' Until out of characters
50936   I.VT9%=I.VT2%+I.VCB%                            ' Offset to character
50938   IF I.VT9%>=2000 THEN I.VER%=11 : GOTO 50950 ' bad field specs
50940   OUT I.RG1%,I.VT9% AND &HFF                      ' Set low address
50942   OUT I.RG2%,I.VT9%\&H100                         ' High address
50946   GOSUB 58000                                     ' Xecute read operation
50948   IF I.VER%=0 THEN 50960                          ' Abort if slave dead
50950     I.VT2%=9999                                   ' Exit loop
50952     GOTO 50970                                    '
50960   I.BUF%(I.VT2%)=CVI(CHR$(INP(I.RG3%))+CHR$(INP(I.RG2%)))
50962   I.VT2%=I.VT2%+1                                 ' Point to next location
50970 WEND
50972 RETURN                                            ' All done
51000 REM Write field                                   '*WRFLD ( I.VER% )I.VCL%,)
51002 I.VER%=0                                          ' Reset error flag
51004 IF I.VCB%<80 THEN I.VER%=11                       ' Illegal address
51006 IF I.VCB%>=2000 THEN I.VER%=11                    ' Too large
51008 IF I.VCB%+I.VT1%>2000 THEN I.VER%=12              ' Field too long for screen
51010 IF I.VER%<>0 THEN RETURN                          ' Quit if bad parameters
51012 OUT I.RG0%,I.CRD%                                 ' Read attribute from screen
51014 OUT I.RG1%,I.VCB% AND &HFF                        ' Low address
51016 OUT I.RG2%,I.VCB%\&H100                           ' High address
51018 GOSUB 58000                                       ' Xecute!
51020 IF I.VER%<>0 THEN RETURN                          ' Abandon if dead slave
51022 IF (INP(I.RG3%) AND &HFE)<>(I.BUF%(0) AND &HFE) THEN I.VER%=13 : RETURN
51024 I.BUF%(0)=I.BUF%(0) OR 1                          ' Set the MDT flag
51026 I.VT2%=0                                          ' First byte of buffer to write
51028 IF (I.BUF%(0) AND &H10)=0 THEN 51052              ' Check for numeric only
51030   I.VT2%=1                                        ' First buffer location to chk
51032   WHILE I.VT2%<=I.VT1%                            ' Do until all chars done
51034     I.VT0%=I.BUF%(I.VT2%) AND &HFF                ' Mask off any previous EAB
51036     IF I.VT0%>=&H20 THEN IF I.VT0%<=&H29 THEN 51048 ' 0-9 is Ok
51038       IF I.VT0%=&H31 THEN 51048                   ' Minus Ok
51039       IF I.VT0%=&H32 THEN 51048                   ' Period Ok
51040       IF I.VT0%=&H35 THEN 51048                   ' Plus Ok
51042     I.VER%=6 : I.VT2%=9999                        ' "Non-numeric" char
51048     I.VT2%=I.VT2%+1                               ' Next character
51050   WEND                                            ' Loop
51052 I.VT2%=0                                          ' Start from the beginning
51054 OUT I.RG0%,I.CWR%                                 ' Set write command
51056 WHILE I.VT2%<=I.VFS%                              ' Until all characters written
51058   I.VT0%=I.VCB%+I.VT2%                            ' Compute address
51060   OUT I.RG1%,I.VT0% AND &HFF                      ' Low address
```

```
51062   OUT I.RG2%,I.VT0%\&H100                 ' High address
51064   OUT I.RG3%,I.BUF%(I.VT2%) AND &HFF      ' Data w/o EAB
51066   GOSUB 58000                             ' Make slave do it.
51068   IF I.VER%<>0 THEN I.VT2%=9999           ' Force exit if slave DOA
51070   I.VT2%=I.VT2%+1                         ' Next location to write
51072 WEND                                      ' Do it again
51074 RETURN                                    ' All done!
51100 REM Get string from buffer                '*GSTR( I.VST$, I.VT1%
51105 I.VER%=0 : I.VT2%=0                       '       I.VT0%          )
51108 IF I.VOO%>1920 THEN I.VER%=12 : RETURN    ' Offset too large
51110 IF I.VOO%<0 THEN I.VER%=12 : RETURN       ' Offset negative
51112 I.VST$=""                                 ' Clear string
51114 WHILE (I.VOO%<=I.VFS%) AND (I.VT2%=0)     '
51116   I.VST$=I.VST$+CHR$(I.TAB%((I.BUF%(I.VOO%) AND &HFF)+&H100))
51118   I.VOO%=I.VOO%+1                         ' Point to next char
51120   IF LEN(I.VST$)=254 THEN I.VT2%=1        ' Overrun
51122 WEND
51126 RETURN
51200 REM Put string in buffer                  '*PSTR( I.VST$, I.VT1%
51202 I.VER%=0                                  '       I.VT0%          )
51204 IF I.VOO%+LEN(I.VST$)-1>I.VFS% THEN I.VER%=12 : RETURN ' Too long
51206 I.VT3%=LEN(I.VST$) : IF I.VT3%=0 THEN RETURN ' Zero strings easy!
51208 I.VT2%=0                                  ' Offset into buffer
51210 WHILE I.VT2%<I.VT3%                       ' While still characters
51212   I.BUF%(I.VOO%+I.VT2%)=I.TAB%(ASC(MID$(I.VST$,I.VT2%+1,1))+&H0)
51214   I.VT2%=I.VT2%+1                         ' Move pointer
51216 WEND
51218 I.VOO%=I.VOO%+I.VT2%                      ' Pointer for next
51220 RETURN                                    ' Transfer complete
51300 REM Read screen absolute                  '*ABSRD ( I.VER%, I.VST$,
51302 I.VER%=0                                  '         I.VS0$, I.VRO%,
51304 I.VT0%=I.VRO%*80+I.VCL%-1                 '         I.VCL%, I.VRR% )
51306 IF I.VT0%<0 THEN I.VER%=11 : RETURN       ' Invalid cursor address
51308 IF I.VT0%>=2000 THEN I.VER%=11 : RETURN   '
51310 IF I.VT0%+I.VRR%>2000 THEN I.VER%=12 : RETURN ' Can't read that much
51312 I.VT1%=0 : I.VST$="" : I.VS0$=""          ' Offset from start
51314 OUT I.RG0%,I.CRD% : I.VT2%=I.VT0%+I.VRR%  ' Set a read commnad
51316 WHILE I.VT0%<I.VT2%                       ' Until all characters read
51318   OUT I.RG1%,I.VT0% AND &HFF              ' Low order address
51320   OUT I.RG2%,I.VT0%\&H100                 ' High order address
51322   GOSUB 58000                             ' Execute read operation
51324   IF I.VER%<>0 THEN I.VT0%=9999           ' Force loop exit
51326   I.VST$=I.VST$+CHR$(I.TAB%(INP(I.RG3%)+&H100)) ' Get char & convert
51328   I.VS0$=I.VS0$+CHR$(INP(I.RG2%))         ' Unmodified EAB
51340   I.VT0%=I.VT0%+1                         ' Next character
51342 WEND
51344 RETURN
51400 REM Read 3278 cursor position             '*CPOS( I.VRO%, I.VCL%,
51402 I.VER%=0                                  '       I.VER%          )
51406 OUT I.RG0%,I.CAC%                         ' Status & cursor read
51408 GOSUB 58000                               ' Start the slave
51410 I.VT0%=(INP(I.RG2%)*&H100)+INP(I.RG1%)    ' Get absolute address
51411 IF I.VER%<>0 THEN RETURN                  ' Dead slave, quit action
51412 I.VRO%=I.VT0%\80                          ' Compute row
51414 I.VCL%=(I.VT0% MOD 80)+1                  ' And column
51416 RETURN
58000 REM Start & wait for slave'!Start & wait for slave
58002 I.VS9$=TIME$ : I.VT9%=0                   ' A simulated stopwatch
58004 OUT I.RST%,0                              ' Start the slave
58006 I.VT8% = INP( I.RAF% ) AND I.MBS%         ' Get slave busy bit
58008 IF I.VT8% = 0 THEN RETURN                 ' Return with command complete
58010 IF I.VS9$=TIME$ THEN 58006                ' Loop until clock ticks
58012 I.VS9$=TIME$ : I.VT9%=I.VT9%+1            ' Three times
58014 IF I.VT9%<3 THEN 58006                    '
58016 I.VER%=1                                  ' Slave timeout
58018 RETURN                                    '
58100 REM Send key scan code from I.VT3%'*Send key scan code
58102 I.VS9$=TIME$ : I.VT9%=0                   ' Make stopwatch
58104 IF (INP(I.RG0%) AND I.MKY%)>0 THEN 58114  ' Key buffer is ready
58106 IF I.VS9$=TIME$ THEN 58104                ' Loop until clock tick
58108 I.VT9%=I.VT9%+1 : IF I.VT9%<4 THEN 58104  ' Time has not run out
58110 I.VER%=9                                  ' Keystroke timeout
58112 RETURN
58114 IF I.VT3%=0 THEN I.VER%=15 : RETURN       ' Invalid scan code
```

```
58115 OUT I.RG0%,I.CKY%                                  ' Keystroke command
58116 OUT I.RG3%,I.VT3%                                  ' Scan code
58118 GOSUB 58000                                        ' Fire up the slave
58120 RETURN                                             ' And we're done!
58200 REM - Verify Row and Column'*Verify ROW/COL
58206 IF (I.VRO%<0) OR (I.VRO%>24) THEN I.VER%=2 : RETURN
58208 IF (I.VCL%<1) OR (I.VCL%>80) THEN I.VER%=3 : RETURN
58210 RETURN
```

Customer Support Information

Introduction

Digital Communications Associates, Inc. makes every effort to ensure that the product you have purchased is of excellent quality in all respects. All hardware and software products have been tested and subjected to strict quality control procedures. Manuals are designed and written to provide you with complete and accurate instructions on how to use your DCA product. You are encouraged to comment on the IRMA manual by filling out and sending in the customer response form included in this section.

If needed, DCA Customer Support personnel are available to assist you from 8:30 a.m. to 8:30 p.m. EST at telephone number 1-800-631-4171. In Georgia call 1-404-442-4470.

The Customer Support section includes the following information:

Limited Product Warranty, The warranty provides a one year limited warranty on IRMA.

Return for Repair Procedure, This section provides instructions on returning hardware units to DCA for repair or replacement.

Warranty Information and Diskette Replacement Policy, This warranty covers your IRMA diskette. It includes a Ninety (90) Day Limited Warranty for diskettes. If your diskette is defective in materials or workmanship, under normal use during the Ninety (90) day warranty period, you may return it to DCA for replacement. You must contact DCA Customer Support before returning a diskette for replacement.

Statement of Copyright Restrictions, These restrictions apply to all DCA product documentation.

Firmware Copyright Restrictions, These restrictions apply to all DCA products that contain firmware, including your IRMA board.

Product Registration Card, This card allows you to register your product with DCA. Fill it out and send it in. This validates your warranty and provides DCA with a way to keep you informed of any product updates that may be made. Postage is prepaid.

Customer Response Form, This card is provided so that you can comment on the IRMA manual.

The following form provides space for entering IRMA product information. Save this form for your records.

Product Name _____

Serial Number _____

Date of Purchase _____

Place of Purchase _____

_____

_____

Limited Product Warranty

Digital Communications Associates, Inc. ("DCA") warrants product hardware to be free from defects in material and workmanship under normal, proper and intended use in its unmodified condition for one year from the date of purchase by the first End User ("Purchaser"). This warranty does not cover normal wear and tear, or damage caused by accident, negligence, vandalism, alteration, abuse, misuse, improper installation, environmental stress, or acts of God.

DCA warrants that the product firmware will conform to DCA's product specifications prevailing at the time of purchase. This

firmware warranty makes no claim of compatibility with equipment or software supplied or to be supplied in the future by others.

DCA's sole obligation under this hardware and firmware warranty shall be to furnish parts and labor for the repair or replacement of the product found by DCA to be defective in material or workmanship during the warranty period. This limited warranty is non-transferrable and applies to the original End User Purchaser of the product and does not apply to subsequent purchasers through resale by the first End User.

Warranty repairs will be performed at the DCA manufacturing facility or DCA designated repair facility. Purchaser should first contact a Service Representative at DCA to arrange for warranty service. Equipment for warranty service should then be returned postpaid to DCA and be accompanied by proof of purchase and a written description of the defect. Upon repair or replacement the equipment will be redelivered by DCA freight prepaid to the Purchaser.

This express limited warranty is in lieu of all other warranties express or implied, statutory or otherwise, including all implied warranties of merchantability and fitness for a particular purpose or use. DCA shall not be liable for any damages sustained by purchaser or any other party arising from or relating to the use or performance of the product, or to any equipment failure, including, but not limited to consequential damages, nor shall DCA have any liability for delays in replacement or repair of related equipment or the DCA product.

Each End User who returns the Product Registration card to DCA shall be notified of product upgrades as they become available during the limited warranty period.

Extended Warranty Services are available through DCA Customer Support.

Return For Repair Procedure
Hardware Units Only

If a hardware unit proves to be defective while still under warranty, you may return it to DCA for repair or replacement. Note that proof of date of purchase must accompany all products returned for repair that are still under warranty.

Before returning any units, contact DCA's Customer Support. Customer Support will attempt to determine the cause of your difficulty. If the board requires repair or replacement, they will provide a Return Authorization number (RA#). There will be no charge for the replacement or repair of boards still under warranty.

Boards that are no longer under warranty can be repaired or exchanged at an additional cost. An extended 2 year warranty may also be purchased at additional cost. Call DCA Customer Support for further details.

The following explains the procedure for returning a unit to DCA.

1. If possible, pack the unit in its original container. If the original container is not available, wrap the unit in *Non-static* material such as newspaper and pack it in a sturdy, cardboard container.

2. Include the following information for all units returned:

   Name
   Address
   City,State,Zip
   Telephone Number
   RA#
   Problem Description

3. All units should be returned by prepaid postage. *DCA will not accept units that have been sent C.O.D.*

4. Mail package to:

   Digital Communications Associates, Inc.
   Attention: Warranty/Repair
   1000 Alderman Drive
   Alpharetta, Georgia 30201

   Any package received without an RA# on the label may be returned unopened.

Diskette Replacement Policy

Digital Communications Associates, Inc. ("DCA") warrants to the original consumer purchaser (the "Purchaser") only, that the magnetic diskette on which the computer program accompanying this DCA Product (the "Product") and limited warranty is recorded will be free from defects in materials and/or workmanship under normal use and service for a period of ninety (90) days from the date the Product is purchased by Purchaser. If, during this ninety (90) day period a defect in the diskette should occur, the diskette may be returned to DCA for replacement. The DCA Product this diskette accompanies must be registered with DCA. Purchaser should call DCA Customer Support for a return authorization and then return diskette, postage prepaid, with authorization to DCA. DCA will not accept diskettes which have been sent C.O.D. or without authorization. Replacement diskettes will be shipped to Purchaser by a method of DCA's choosing. If Purchaser desires a specific form of conveyance, Purchaser must bear the cost of shipment.

The Purchaser's sole and exclusive remedy in the event of a defect is expressly limited to replacement of the diskette. Purchaser shall be solely responsible for the failure of any diskette resulting from accident, abuse or misapplication of the diskette and DCA assumes no liability as a consequence of such events under the terms of this Limited Warranty.

Any direct, incidental or consequential damages, such as, but not limited to, loss of anticipated profits or benefits, caused by malfunction, defect or otherwise and with respect to breach

of any express or implied warranty are not the responsibility of DCA or anyone else who has been involved in the design, production or delivery of the diskette on which the Program is recorded and to the extent permitted by law, are hereby excluded both for property and to the extent not unconscionable, personal injury damage.

This Limited Warranty gives Purchaser specific legal rights, and Purchaser may also have other rights which vary from state to state.

Statement of Copyright Restrictions

The Digital Communications Associates, Inc. ("DCA") manual you have purchased is copyrighted by DCA and all rights are reserved. Your rights of ownership are limited and restricted by the copyright laws of the United States.

It is against the law to copy or in any way reproduce or translate any part of this manual without the express written permission of DCA.

This is not a full statement of copyright laws. For a complete statement of the restrictions imposed on you under the copyright laws of the United States, see Title 17, United States code.

Firmware Copyright Notice

This IRMA™ product contains firmware which has been copyrighted by Digital Communications Associates, Inc., ("DCA"). All rights are reserved. *Your rights of ownership are subject to the limitations and restrictions imposed by the copyright laws (Title 17, United States code).*

It is against the law to copy, reproduce, or transmit (including, without limitation, electronic transmission over any network) any part of the firmware.

This is not a full statement of copyright laws. For a complete statement of the restrictions imposed on you under the copyright laws of the United States, see Title 17, United States code.

Reader Response Form

In order to better serve our customers' needs, we welcome comments or questions regarding our publications. Please take the time to fill out the following questionnaire.

Name of DCA Product: _____

Is this manual:	Poor	Fair	Good	Excellent
Easy to read?	☐	☐	☐	☐
Easy to follow?	☐	☐	☐	☐
Easy to understand?	☐	☐	☐	☐
Detailed enough?	☐	☐	☐	☐
Too detailed?	☐	☐	☐	☐
Are technical terms clearly defined?	☐	☐	☐	☐
Are the illustrations helpful?	☐	☐	☐	☐

	Yes	No
Did you find any inconsistencies or errors in the instructions?	☐	☐

If yes, please describe:

_____

_____

_____

	Yes	No
Do the instructions provide all the information needed to use the product?	☐	☐

If not, what additional information do you feel is needed?

_____

_____

_____

NAME (optional): _____

TITLE: _____

COMPANY: _____

ADDRESS: _____

CITY: _____ STATE: _____ ZIP: _____

Thank you for taking the time to assist us. Return to:

Manager of Technical Publications
Digital Communications Associates, Inc.
1000 Alderman Drive
Alpharetta, Georgia 30201

IMPORTANT REGISTRATION INFORMATION

Please help us keep your IRMA™ up-to-date by registering with us immediately! This update service is provided FREE during IRMA's warranty period.

Thank you.

IRMA Product Registration Card

Contact Name: _____

Title: _____

Company: _____

Street Address: _____

City: _____ State: _____ Zip: _____

Country: _____ Telephone: / /

IBM Controller Type: _____ Model: _____

IBM Processor Type: _____ Operating System: _____

PC Type(s): IBM PC
to be used with this unit _____ Other(s): _____

PC Diskette Type: Single Sided _____ Double Sided

Operating System(s) in use:
on these PC's _____

Comments: _____

_____

IRMA Serial Number: _____ Date Installed: _____

3/85-I

Digital Communications Associates, Inc.
1000 Alderman Drive, Alpharetta, Georgia 30201

MIC-016B
Printed in U.S.A.

40-97915-900